НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Олександр Коваль

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення розподілених систем»

спеціальності 121 - «Інженерія програмного забезпечення»

на тему: «Автоматизована система збирання, кодування та зберігання облікових даних ургентних та планових пацієнтів лікарні»

Виконав:
студент IV курсу, групи ТВз-61
Сушко Дмитро Анатолійович _____

Керівник:
Доцент, кандидат технічних наук, доцент,
Крячок Олександр Степанович _____

Консультант з _____:
_____
_____ _____

Рецензент:
Доцент, кандидат технічних наук, доцент,
Реуцький Микола Олександрович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки: 121 – «Інженерія програмного забезпечення»

Спеціалізація: «Програмне забезпечення розподілених систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020 р.

ЗАВДАННЯ
на дипломну роботу студенту

_____Сушко Дмитро Анатолійович_____
(прізвище, ім’я, по батькові)

1. Тема роботи Автоматизована система збирання, кодування та зберігання облікових даних ургентних та планових пацієнтів лікарні_____
_____

керівник роботи __Крячок Олександр Степанович, кандидат технічних наук_
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від 25 травня 2020 р. № **1168-с**

2. Строк подання студентом роботи 12 червня 2020 року_____
3. Вихідні дані до роботи мова програмування Java, графічна бібліотека Swing, база даних Microsoft SQL Server, фреймворк Selenium_____
_____

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Розробити автоматизовану систему збирання, кодування та зберігання облікових даних пацієнтів лікарні що може заповнювати анкети госпіталізацій пацієнтів лікарні на сайті UDRG-System.com, дублювати відправлені дані зберігаючи їх у локальній базі даних, дозволяти переглядати збережені раніше дані, виявляти помилки при заповненні анкет.
_____
_____
_____
_____

5. Перелік ілюстративного матеріалу
Діаграма прецедентів, Діаграма послідовностей, Зображення інтерфейсу системи та окремих елементів_____
_____
_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 4 Листопада 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	11.11.2019	Виконано
2.	Вивчення та аналіз задачі	18.11.2019	Виконано
3.	Розробка архітектури та загальної структури системи	15.01.2020	Виконано
4.	Розробка структур окремих підсистем	14.02.2020	Виконано
5.	Програмна реалізація системи	13.03.2020	Виконано
6.	Оформлення пояснювальної записки	15.04.2020	Виконано
7.	Захист програмного продукту	11.05.2020	Виконано
8.	Передзахист	10.06.2020	Виконано
9.	Захист		

Студент _____ _____Сушко Д.А._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Крячок О.С._____
(підпис) (прізвище та ініціали,)

АННОТАЦІЯ

Дипломну роботу на тему «Автоматизована система збирання, кодування та зберігання облікових даних ургентних та планових пацієнтів лікарні» виконано на 75 сторінках, що включають 15 ілюстрацій, додаток та 30 бібліографічних посилань.

Метою цієї роботи є розробка автоматизованої системи збирання, кодування та зберігання пацієнтів лікарні яка б дозволила полегшити виконання своїх службових обов'язків лікарям під час заповнення анкет пацієнтів на сайті UDRG-System.com (державний сайт для розрахунку вартості медичних послуг та вагових коефіцієнтів в закладах охорони здоров'я України) та мати змогу отримувати більше користі від ретельно заповнених анкет пацієнтів.

Для досягнення поставленої мети, серед лікарів було зібрано інформацію про їх нагальні проблеми пов'язані з заповненням анкет на сайті UDRG-System.com, проаналізовано існуючі системи для роботи з даними пацієнтів лікарень, спроектовано архітектуру та дизайн системи, обґрунтовано використання фреймворків та бібліотек, розроблену систему автоматизованого збирання та зберігання пацієнтів лікарні.

Ключові слова: система електронного документообігу, атестаційна дипломна робота, MS SQL, Swing, Selenium.

Abstract

Qualifying work on the topic “Automated system for collecting, processing and storage of medical data of urgent and scheduled hospital’s patients” is performed on the 75 pages, which includes 30 illustrations, appendix and 30 bibliographic references.

The purpose of this work is to develop such automated system for collecting, processing and storage of medical data of hospital’s patients that would improve the ability of medical personnel to commit their duties when they need to fill information about patient’s hospitalization on UDRG-System.com (governmental website for calculating cost of medical services in medical institutions of Ukraine) and to allow using previously filled data to improve quality of medical services.

To achieve this goal, the information about usage of UDRG-System.com and problems they are facing while doing it was gathered amongst medical workers, the analysis was conducted on existing automated systems for processing medical data, the architecture of the future system and its design were created, the usage of specific frameworks and libraries was justified and system itself was developed.

Keywords: medical document management system, MS SQL, Swing, Selenium.

Зміст

ВСТУП

У сучасному світі все більшого значення набуває електронний документообіг. На відміну від паперових документів, електронні документи мають ряд значних переваг. Насамперед це зручність та надійність їх зберігання. Так історично склалося що через складнощі зберігання було ненавмисно знищено дуже багато важливих документів. Електронні документи не мають подібної проблеми. Їх дуже легко зберігати та копіювати у разі необхідності не боячись втратити частину інформації при копіюванні. Сучасні сервери та хмарні технології використовують велику кількість технологій запобігання втрати інформації.

Наприклад такі як RAID-масиви (технологія віртуалізації зберігання даних, що поєднує в собі кілька компонентів фізичного дискового накопичувача для надмірності даних) або Version Control System (Системи для відстеження змін та ведення журналу подій). Іншою важливою особливістю є можливість доступу до електронних документів яка не обмежена відстанню. Людина, на іншій стороні земної кулі, може легко прочитати будь-який електронний документ без необхідності бути фізично присутнім у місці його зберігання. Але цим переваги електронних документів не обмежуються. На відміну від паперових документів їх також можливо змінювати. Це значно пришвидшує документообіг. Саме тому не дивно що однією з основних галузей застосування електронного документообігу в світі є медицина. Усі зазначені вище переваги мають велике значення у сфері охорони здоров’я.

Якісна система електронного документообігу дозволяє лікарям своєчасно отримувати інформацію про хворих, додавати нові записи до медичних карток або редагувати старі. Більш того, ця система може бути розширена та втілювати ще більший функціонал - наприклад сканування та збереження старих медичних документів, можливість лікарям обговорювати між собою діагнози та результати аналізів, систему оповіщення про надзвичайні випадки.

Як показує досвід застосування інформаційних систем медичним центром Mount Sinai (США)[1] впровадження подібної системи дозволило зменшити кількість повторних звернень хворих на 56 відсотків за що вони отримали престижну у галузі медицини премію Davies Award.

Що саме стало запорукою їх успіху? По-перше це можливість створювати математичні моделі на основі карток пацієнтів за допомогою яких, стає можливим передбачити появу ускладнень та повторних звернень у майбутньому. По-друге зменшилась кількість роботи з паперами у кожного лікарям, тим самим звільнивши більше часу для виконання своїх основних функцій. По-третє через те, що всі дані пацієнта стали знаходитися у одному місці, новим лікарям стало легше виконувати свої обов’язки тому що вони почали мати повний доступ до всіх медичних даних хворого, якого вони лікують.

1 ОСОБЛИВОСТІ ПОБУДОВИ СИСТЕМ ДЛЯ РОБОТИ ЗА ДАНИМИ ПАЦІЄНТІВ

1.1 Переваги застосування автоматичних систем для роботи з даними пацієнтів лікарні

Щоб краще розуміти яким саме чином треба реалізувати автоматизовану систему, спочатку треба розглянути які саме задачі вона може виконувати. Подібні системи вже достатньо довгий час застосовуються у США та країнах Європи і на основі цього досвіду вже сформовано декілька напрямків застосування.

Прогнозування очікуваної кількості пацієнтів. Аналіз оцифрованих медичних даних можна використати для вирішення однієї з суттєвих проблем у галузі охорони здоров'я, з якою стикаються тисячі керівників лікарень. Щороку багато пацієнтів помирають через недоступність лікаря в найкритичніший час. Аналіз зібраних медичних даних може дозволити з відносною похибкою передбачати кількість лікарів, необхідних для найбільш ефективного обслуговування пацієнтів та оптимально зосередити людський ресурс лікарні. Чим більше зібрано медичної інформації тим точнішим буде прогноз. А це дозволить скоротити час очікування для пацієнтів та підвищити якість медичних послуг.

Електронні медичні записи. Електронні медичні записи - це одне з найкращих рішень у галузі охорони здоров'я. На ранніх етапах надання медичної допомоги створювало серйозну проблему реплікації даних. Кожні клініка мала свою паперову базу даних, де вона зберігала медичні картки пацієнтів. Щоб отримати інформацію про історію хвороб пацієнта, лікар мав зв'ятися з клінікою де хворий лікувався раніше. Цю проблему було вирішено через створення єдиної бази даних для медичних карток.

Там зберігаються важливі дані пацієнтів, що включають історію хвороби та загальну інформацію, які доступні лише для авторизованих користувачів, таких як організації охорони здоров'я, уряд та лікарі. Також було забезпечено надійне збереження даних та прийняті заходи для запобігання будь-якого несанкціонованого доступу.

Єдина база медичних карток дозволяє створювати електронні статистичні звіти, що містять демографічні показники, історію алергій, медичні тести або результати огляду здоров'я всіх пацієнтів. Також її використання дозволяє нагадувати пацієнтам, якщо вони потребують медичного тестування або якщо вони не дотримуються вказівок лікаря. Це зменшує ризи медичних ускладнень, дозволяючи людям відслідковувати свій процес лікування та історію хвороб.

Сповіщення в режимі реального часу. Інформаційні технології здатні принести користь як людям, так і суспільству, та зменшити ризик втрати життя людей. Основна ідея полягає у тому, щоб почати лікування ще до того як людина почне страждати. Дуже багато людей помирає в результаті надходження в лікарню надто пізно. Отже, що цьому завадити, програмний застосунок може відстежувати будь-якого пацієнта в режимі реального часу та ділитися необхідними даними з лікарями, щоб вони могли вжити заходів ще до того, як ситуація стане критичною.

Також цей застосунок може збирати дані про стан здоров’я пацієнта за допомогою портативних пристроїв для більш ефективного контролю. Усі зібрані дані зберігаються у хмарному сховищі та аналізуються за допомогою спеціальних комп’ютерних програм. Якщо помічена якась надзвичайна активність, система може автоматично оповіщати відповідний персонал лікарні. Наприклад якщо будь-який пацієнт стикається з проблемами високого кров'яного тиску або астми, застосунок відправляє повідомлення лікарям. Окрім цього, зібрані застосунком статистичні дані можуть бути використані задля науки та вдосконалення процесу лікування конкретних захворювань.

Підвищення залученості пацієнтів. Це використання функціоналу портативних приладів для відстеження стану здоров'я та передбачення захворювань, від яких пацієнт може страждати в майбутньому. Інформаційна система пов'язує результати, отримані від медичних пристроїв, з іншими відстежуваними даними, щоб виключити ризик потенційними захворювань. Крім того, це також допомагає лікарю виявити симптоми певних захворювань для надання кращого обслуговування.

Це застосування зосереджується на використанні важливих медичних показників, які збираються у пацієнтів за допомогою портативних приладів, що відстежують стан здоров'я, таких як частота серцевих скорочень, артеріальний тиск тощо. Завдяки цьому з’являється можливість залучати людей до покращення медичного обслуговування та використовувати аналітику даних для виявлення симптомів. Зібрані дані від пацієнтів зберігаються на сервері, де лікарі можуть перевірити, чи є стан здорово’я пацієнта нормальним, і надати відповідну консультацію. Пацієнтам, які страждають від підвищеного артеріального тиску, астми, мігрені чи інших серйозних проблем зі здоров’ям, лікарі допоможуть спостерігати за своїм способом життя та вносити зміни, якщо це важливо. Мета цього застосування - зменшити частоту відвідування лікарів через незначні проблеми, регулюючи щоденну діяльність.

Запобігання зловживанню опіоїдами. У світі існує серйозна проблема надмірного вживання опіоїдів і тому не дивно що саме тут інформаційні системи можуть бути у нагоді для галузі охорони здоров’я. Використовуючи техніку fuzzy logic для виявлення 742 факторів ризику, які можна використати для прогнозування того, чи зловживає хворий опіоїдами. Збираючі дані від страхових компаній та аптек та поєднуючи їх зі статистичним аналізом дає змогу створити досить точний прогноз.

Це дозволяє не тільки визначити пацієнтів, які зловживають опіоїдами, але й повідомляти проце медиків. Завдяки використанню алгоритму Forest дає можливість для запобігання несвідомого передозування опіоїдами.

Стратегічне планування з використанням даних про здоров'я пацієнтів. Це застосування використовує дані, пов’язані зі здоров’ям, щоб надихнути людей відвідувати медичну організацію для лікування. Система збирає різні види даних, які включають демографічні показники, кількість населення, результати перевірок тощо. Проаналізувавши великий об'єм даних, вона використовує результат для стратегічного планування для виконання певної діяльності. Наприклад це дозволяє отримувати наукові дані для виявлення проблем, яких не помітно на перший погляд. Проводити оцінку поведінки пацієнта, проаналізувавши теплову карту їх розташування. Визначати причини таких явищ, таких як швидкий приріст населення чи поширення будь-яких епідемічних захворювань. Сповіщати відповідний персонал, чи слід оновити процес лікування після аналізу результату підходу, орієнтуючись на проаналізовані дані. Вираховує необхідну кількість лікарень або медичних служб, це дозволяє приймати рішення, де саме необхідне створення нових медичних організацій.

Лікування раку. Рак - це захворювання, яке не має специфічного лікування викликане через аномальний ріст клітин. Це застосування одне з найбільш перспективних ініціатив, прийнятих на сьогодні, яке використовує зібрані дані для пошуку вирішення серйозного захворювання. Це застосування аналізу даних пацієнтів для того, щоб підібрати найкращі методи для лікування раку. Цей проект все ще знаходиться в процесі розробки але воно може прояснити механізми захворювання та лікування інших небезпечних хвороб.

Найбільшим викликом для цього застосування є обробка інформації зібранної з багатьох різних джерел та поєднання цих наборів даних один з одним. Це дозволяє зібрати всі попередні звіти про біопсії для того щоб лікарі

отримали максимум інформації, перш ніж приймати рішення. Цей метод вже допоміг встановити що антидепресант Дезипрамін працює як ліки від деяких видів раку легенів. Усе це дозволяє лікарям порівнювати інформацію надану системою охорони здоров’я, щоб визначити найбільш відповідну до даного випадку та призначити правильне лікування.

Прогнозування та аналітика в галузі охорони здоров'я. Це автоматизований інструмент для роботи з великими об'ємами даними у галузі охорони здоров'я, який допомагає лікарю призначити ліки пацієнтам протягом лічених секунд. Він використовує електронні медичні записи, зібрані від багатьох страхових компаній, лікарень, діагностичних центрів та громадських медичних центрів. Це дає змогу легко визначити, чи існує загроза для пацієнта страждати від захворювання в майбутньому. Окрім цього, база даних, що містить чутливі дані, може бути додатково використана для вдосконалення процесу охорони здоров’я.

Цей підхід орієнтований на те щоб зменшити ризик помилки лікарів під час призначення лікування завдяки використанню реляційної бази даних та інструментів прогнозування та аналітики. Наприклад деякі пацієнти мають дуже незвичну медичну історію і ця програма дозволить медикам добре лікувати таких пацієнтів. Найкращою перевагою цієї програми є те, що вона дозволяє передбачити, чи є у пацієнта високий ризик діабету та інших хронічних захворювань.

Телемедицина. Незважаючи на те, що ця концепція існує вже багато років у сфері охорони здоров'я, вона побачило світло лише після поєднання з обробкою великих об’ємів даних, смартфонами та портативними пристроями. Таке застосування забезпечує можливість надання медичної допомоги віддалено за допомогою технологій. Воно перш за все призначене для первинних методів лікування та можливості дистанційно стежити за критичними пацієнтами. Крім того воно:

- Дозволяє лікарям здійснювати операції віддалено з доставкою даних у режимі реального часу.
- Допомагає відстежувати стан пацієнта, регулюючи схеми його лікування та запобігаючи погіршенню стану здоров'я.
- Оцифровує процес лікування, оскільки пацієнти можуть приймати поради лікарів у будь-який час і в будь-якому місці. Оскільки стан здоров'я пацієнта таким чином легше контролювати, це економить багато часу для пацієнтів та забезпечує ефективне надання медичної допомоги.

Окрім цього, телемедицина також пропонує медичну освіту для професіоналів.

Обробка графічних медичних даних. Ще кілька років тому це здавалось неможливим, але сучасна наука вирішила одну із суттєвих проблем у галузі охорони здоров’я, яка полягає у збереженні графічних медицинських даних у точному вигляді. Медичні зображення є необхідними рентгенологам для будь—якого виявлення захворювань. Це застосування дозволяє замінити зображення цифрами та використовувати на них алгоритми для більш якісного збору медичної інформації.

Колись цей метод може повністю замінити людей-рентгенологів машинним алгоритмом. Більш того, цей алгоритм буде якісніше обробляти інформацію ніж це дозволяє людське оке. Він також може бездоганно розкривати вказані закономірності, пов'язані з патологією, обчислити стан кісток і передбачити, чи загрожує пацієнту перелом чи ні.

Уже зараз використання таких алгоритмів підвищує ефективність роботи рентгенологів. Завдяки цьому процесу рентгенолог може оглянути набагато більше знімків, ніж раніше.

Запобігання надто частим потребам у невідкладній допомозі. Цей напрямок фокусується на економії грошей та часу пацієнта, використовуючи аналітику медичних даних у галузі охорони здоров'я. Основними пріорітетами є:

- Зменшення витрат грошових коштів платників податків та організацій охорони здоров'я та забезпечення надання кращої допомоги хворим.

- Аналіз причин повторних звернень та їх попередження.

- Допомога медичним страховим компаніям надати найкращу послугу та полегшити виявлення шахрайских дій.

- Коли пацієнту потрібно кілька разів заплатити за одне і те ж медичне обстеження, це спричиняє марну трату грошей і тому є потреба у запобіганні подібним ситуаціям.

- Ведення обліку лікування, яке вже отримав один пацієнт, та яке може бути перевірено консультантами, перш ніж приймати рішення.

- Здійснює доступ до даних для місцевих медичних працівників, які зберігаються в базі даних для розслідування використання відділення екстреної допомоги, прийому лікарні та запобігання повторному зверненню.

Зменшення шахрайства та підвищення безпеки. З тих пір, як з'явилася ідея медичного страхування, постачальники послуг стикаються з серйозною проблемою помилкових претензій та забезпеченням кращого обслуговування пацієнтів, які дійсно потребують допомоги. Крім того, загрози копіювання даних та маніпуляції конфіденційними стали дуже поширеними у нас час. Це застосування інформційних технологій захищає цінні дані багатьох пацієнтів від злочинців, які можуть продати їх на чорному ринку.

Кіберзагрози у мережі Інтернет є великою проблемою для компаній, що збирають дані. Допомога підприємствам, які працюють з критичними та

конфіденційними даними, захищаючи їх від загрози безпеці є критично важливою. Застосування автоматизованих систем для роботи з даними пацієнтів може дозволити:

- Успішно виявляти шахрайство та надавати медичні послуги саме тим людям, які цього потребують.
- Охороняти цінні дані, щоб запобігти їх потраплянню у чужі руки, звідки злочинці можуть використовувати їх для створення небажаних ситуацій.
- Крім того це долзволить забезпечити надійне виявлення недостовірних претензій та економити багато грошей для держави та страхових компаній щороку.

Трансформація допомоги хворим на діабет. Щороку дуже багато людей стають хворими на діабет, цих випадків настільки багато що з часом вони можуть досягти рівню епідемії. Це одна з 7 головних причин, що призводять до проблем, пов'язаних зі здоров’ям. Застосування автоматизованих систем обробки медичнох інформації дозволяє збирати поведінкові, фізіологічні та контекстуальні дані від пацієнтів, щоб оцінити, використовуючи спеціальні алгоритми, та поліпшити догляд за хворими на діабет.

Дані збираються за допомогою переносних цифрових пристроїв, таких як вимірювачі рівня глюкози в крові, манжети артеріального тиску та ваги. Зберігання даних у доступній базі даних також є частиною цього застосування. Оцінка даних для отримання потенційної інформації про спосіб життя пацієнта та можливість зворотнього зв'язку, позволяє повідомити пацієнтів у необхідності зміни способу життя на більш здоровий.

Основними перевагами є:

- Автоматизація процесу доставки інсуліну.

- Використання closed-loop системи щоб зрозуміти, як користувач реагує на їжу, фізичні вправи та інсулін.
- Розуміння поточного стану здоров'я пацієнта та можливість його сповіщення ще до того як проблеми зі здоров'ям стануть невідворотними.

Прогнозування серцевих нападів. Серцевий напад - одна із найбільш смертоносних проблем зі здоров’ям, яка спричиняє втрату багатьох життів щороку. Прогнозувати непередбачувані інфаркти непросто і це вимагає збору великих об'ємів медичних даних. Крім того, щоб мати можливість передбачити ймовірність гострого інфаркту треба мати змогу порівнювати, встановлювати взаємозв'язки між наборами даних та застосовувати ці дані для знаходження скритих закономірностей. Подібні системи відстежують поточні тенденції у зміні здоров'я пацієнта та повідомляють про необхідність вжити заходів.

Цей напрямок призначений для оцінки складних наборів даних для прогнозування, попередження, управління перебігом та лікування захворювань, пов’язаних із серцем, таких як інфаркти. Для цього він вивчає величезні національні та міжнародні бази даних шоб досягти мети та дати найкращі результати. Аналізуючи харчові звички користувача, спосіб життя та записи рецептів, можна передбачити, чи загрожує йому ризик будь-яких серцево-судинних захворювань. Відстеження записів, зібрані з переносних пристроїв, які можуть обчислити потік клітин крові, серцебиття, артеріальний тиск, також покращує передбачення можливості інфаркту в майбутньому.

1.2 Існуючі системи

Розглянемо найбільш відомі у світі системи електронного документообігу для лікарень.

1.2.1 CRM система Kapture.

Kapture Healthcare CRM - це платформа автоматизації, призначена для медичних клінік та лікарень. Це дуже легко налагоджуємий інструмент, який адаптується до вимог будь-якої галузі чи бізнесу. Це допомагає фіксувати потенційних клієнтів, керувати відвідуваннями та записами пацієнтів - і все це в одній єдиній платформі CRM. Крім того, це програмне забезпечення для охорони здоров'я забезпечує простий користувальницький інтерфейс і легко інтегрується в існуючу інфраструктуру будь-якої організації. Підтримує понад 500 API, що дозволяє легко синхронізувати велику організацію. Крім того, цей програмний застосунок полегшує користувачам імпорт або експорт даних у різних форматах. Більше того, ці дані також можуть використовуватися як вхідні дані для інших існуючих систем.

1.2.2 CRM система Dquip.

Dquip - це програмне забезпечення для управління відносинами з клієнтами, розроблене для того, щоб допомогти організаціям з продажами, керувати консультаціями, ефективно стежити за пацієнтами, оптимізувати всі робочі процеси та сприяти зростанню. Цей CRM додаток працює бездоганно на останніх смартфонах, настільних комп'ютерах та планшетах, оскільки він дуже гнучко налагоджується. Крім того, програмне забезпечення побудоване на таких технологіях з відкритим кодом, як LAMP, що розшифровується як Linux, Apache, MySQL та PHP. Воно також інтуїтивно працює на всіх останніх версіях операційних систем Windows, Linux та MacOS. Підходить для стоматологічних клінік, очних клінік, фізіотерапевтичних центрів, медичних лабораторій та діагностичних центрів, фітнес-центрів, тренажерних залів, курортів тощо.

1.2.3 CRM система Bpm'online.

Bpm'online - це інтелектуальна платформа CRM, яка впорядковує всі робочі процеси, створюючи єдину базу даних для всіх контактів та облікових записів, з якими їй доводиться працювати. Ця CRM в охороні здоров'я включає в себе суттєві функції, такі як інтегровані записи пацієнтів, історія взаємодій та

перелік заходів, у яких пацієнти беруть участь. Програмне забезпечення Bpm’online значно полегшує управління контентом підприємства та системами бухгалтерського обліку. Крім того, цей програмний застосунок може витягувати інформацію про клієнтів та ефективно її використовувати. Він також використовує електронні записи для автоматизації процесів реєстрації, догляду за пацієнтами та виставлення рахунків.

Однак у всіх вищезгаданих CRM-системах є великий недолік – цінова політика. Вони розраховані більше на використання у приватних лікарнях ніж у державних медичних закладах і тому мають велику ціну за обслуговування. Варто відмітити що послуги, які вони надають теж дуже якісні.

1.2.4 Електронна Медична Система Helsi.me

В Україні однією з найбільш поширених систем електронного документообігу у медицині є Helsi.me. Її особливістю є орієнтованість на пацієнта. На жаль це не завжди задовольняє потреби лікарів. На даний момент існує необхідність у системі документообігу яка зможе спростити роботу лікарів та, у деяких випадках навіть пришвидшити її. Обов’язково потрібна можливість додавання нових функцій, яких будуть потребувати медичні робітники. Для цього треба постійно отримати зворотній зв’язок від майбутніх користувачів. Розуміючи що саме потрібно від нової системи документообігу, буде можливим створити саме такий інструмент, який дозволить зробити роботу медпрацівників більш ефективнішою. Однією з найбільш потрібних функцій, якою нема в Helsi.me це сумісність з UDRG-System.com (державний сайт для розрахунку вартості медичних послуг та вагових коефіцієнтів в закладах охорони здоров`я України). Тому забезпечення цієї вимоги є найбільшим пріоритетом.

1.3 Існуючі кіберзагрози автоматизованим системам збору та збереження медичних даних

Лише за один 2019 рік було здійсненно безліч атак на медичні установи та вкрадено величезні архіви інформації. Ось лише деякі з них:

Amca Data Breach. На початку травня подання до Комісії з цінних паперів та бірж виявило, що постачальник послуг з виставлення рахунків Американське агентство медичних зборів було зламане протягом восьми місяців між 1 серпня 2018 року та 30 березня 2019 року.

З моменту виявлення порушення, щонайменше шість охоплених організацій виступили з повідомленням про крадіжку даних своїх пацієнтів. Однак більшість постачальників постраждалих все ще продовжують розслідувати це порушення, тому загальна кількість постраждалих пацієнтів буде незрозумілою в осяжному майбутньому.

На даний момент постраждало до 12 мільйонів пацієнтів з Quest Diagnostics. Зламана система включала особисту та фінансову інформацію гіганта тестування, включаючи номери соціального страхування та медичну інформацію. До 7,7 мільйона пацієнтів з LabCorp також були потенційно уражені, а також 422 000 пацієнтів BioReference. Нещодавно до підрахунку були додані ще дві охоплені особи: Центр охорони здоров'я Пенобско в штаті Мен з 13000 хворими, які постраждали, та Лабораторії клінічної патології з 2,2 мільйонами пацієнтів. Шостий постачальник, Austin Pathology Associates, повідомив, що принаймні 46 500 його пацієнтів зазнали події. Невдовзі ще сім охоплених організацій повідомили, що вони теж зазнали впливу: Natera, American Esoteric Laboratories, CBLPath, South Texas Dermatopathology, Seacoast Pathology, Arizona Dermatopathology, and Laboratory of Dermatopathology ADX. Загалом було вкрадено дані понад 774 640 пацієнтів (Natera не розкрила, дані скількох пацієнтів зазнали крадіжки), що призвело до загальної кількості у понад 25 мільйонів постраждалих.

Батьківська компанія AMCA з тих пір зареєструвала банкрутство, тоді як постачальник платних послуг, Quest та LabCorp, стикаються з численними розслідуваннями та судовими процесами

Dominion National. Страховик Dominion National повідомив про дев'ятирічний взлом на своїх серверах, що потенційно зачепило дані 2,96 мільйона пацієнтів.

Внутрішня тривога виявила несанкціонований доступ до систем, що викликало розслідування. Чиновники заявили, що встановлено, що несанкціонований доступ розпочався ще 25 серпня 2010 року, майже за дев'ять років до того, як порушення було виявлено у квітні 2019 року.

Сервери містили інформацію про зарахування нових клієнтів та демографічну інформацію про нинішніх та колишніх членів страхового плану Dominion National, а також дані про стоматологічні та зорові проблеми людей. Дані постачальників медичних послуг також були поставлені під загрозу.

Inmediata Health Group. Неправильно налаштована база даних призвела до крадіжки особистих даних щодо здоров’я 1,57 мільйонів пацієнтів Inmediata Health Group. Що ще гірше: постачальник ненавмисно відправив пацієнтам неправильні листи під час повідомлення про порушення.

Компрометована база даних була виявлена в січні, коли чиновники виявили, що функція пошукової системи дозволила індексувати внутрішні веб-сторінки Inmediata, що використовуються для бізнес-операцій. В результаті було виявлено деяку електронну інформацію про здоров'я.

Дослідження визначало демографічні дані пацієнтів, дані медичних претензій та іншу особисту інформацію, конфеденційність якої була потенційно порушена. Але коли Inmediata надсилала сповіщення пацієнтам про інцидент із безпекою, деякі пацієнти повідомили, що отримують декілька листів, деякі з

яких були адресовані до інших пацієнтів. Генеральний прокурор штату Мічиган досі розслідує інцидент.

UW Medicine. У лютому Університет Медицини Вашингтону почав сповіщати 974 000 пацієнтів, що їхні дані були викриті в Інтернеті протягом трьох тижнів через неправильно налаштований сервер. Порушення було виявлено у грудні 2018 року, коли пацієнт провів пошук власного імені та знайшов файл із своїми особистими даними. Вони повідомили UW Medicine, яка визначила помилку працівника за три тижні до того, як внутрішні файли стали загальнодоступними.

"Оскільки компанія Google зберегла деякі файли до 26 грудня 2018 року, компанія UW Medicine працювала з Google, щоб видалити збережені версії та не допустити їх відображення в результатах пошуку", - заявили чиновники. "Усі збережені файли були повністю видалені з серверів Google до 10 січня 2019 року." База даних містила особисту інформацію, включаючи назву лабораторного тесту або медичного дослідження з описом стану здоров'я деяких пацієнтів.

Wolverine Solutions Group. Системи WSG були заражені програмою, яка вимагає викупу, у вересні, а дешифрування та відновлення файлів тривали впродовж жовтня. Кібератака потенційно поставила під загрозу широкий спектр даних багатьох клієнтів, включаючи демографічні дані та номери соціального страхування.

Департамент звітності про порушення охорони здоров'я та соціальних служб показав, що WSG повідомила про постраждалі 48 471 пацієнта. Однак, генеральний прокурор штату Мічиган Дана Нессел стверджує що за підрахунками, постраждало щонайменше 600 000 жителів Мічигану, і в Health Alliance Plan було повідомлено, що дані 120 000 пацієнтів опинились під загрозою.

Blue Cross Blue Shield of Michigan, Three Rivers Health, North Ottawa Community Health System, Mary Free Bed Rehabilitation Hospital, Covenant Hospital, Sparrow Hospital, and McLaren Health Care, також повідомили, що їхні пацієнти постраждали від інциденту.

Ця проблема не оминула і Україну. 27 липня 2017 року неочікувана кібератака пошкодила численні комп'ютерні системи в Україні та інших країнах. Атака була здійснена вірусом, який ESET визначив як Diskcoder.C (він же ExPetr, PetrWrap, Petya або NotPetya). Ця атака маскувалася під епідемію звичайного шифрувальника – який шифрував дані на диску і вимагав 300 $ в біткоінах для відновлення даних. Але насправді, план був в нанесенні збитку, тому автора зробили все що могли, щоб ускладнити розшифровку даних. На початковій фазі поширення вірусу DiskCoder.C, було використано популярне програмне забезпечення бухгалтерського обліку M.E.Doc - практично монополіст на Україні в області передачі звітності в фіскальні служби. Ця атака завдала великих збитків економіці країни.

Отже, як ми бачимо на прикладі бухгалтерської системи електронного обороту M.E.Doc, зараз як ніколи важливим є забезпечення безпеки даних. Медичні дані пацієнтів є не менш важливою інформацією. "Злита" база даних, яка містить у собі історію хвороб, може використовуватись як для шантажу так і для руйнації репутації пацієнтів лікарень.

1.4 Засоби розробки

1.4.1 Застосування мови програмування Java

Для написання цієї системи було обрано мову програмування Java. Java - це мова програмування високого рівня, розроблена компанією Sun Microsystems. Спочатку вона планувалася для написання програм для приладів та портативних пристроїв, але згодом стала популярним вибором для створення серверної частини веб-додатків. Синтаксис Java схожий на синтаксис C ++, але є строго об`єктно-орієнтованою мовою програмування. Наприклад, більшість, якщо не всі, програми на Java містять класи, які використовуються для визначення об`єктів, та методи, які призначаються окремим класам. Java також відома тим, що є більш строго типізованою, ніж C ++, тобто змінні та функції повинні бути чітко визначені. Це означає, що сама по собі Java обмежує типи помилок, які можуть бути викликані невизначеними змінними або непризначеними типами.

Код, написаний на Java, надзвичайно портативний. Один й той самий програмний застосунок на Java працюватиме однаково на будь-якому комп’ютері, незалежно від апаратних особливостей або операційної системи, якщо у ній є інтерпретатор Java. На відміну від виконуваних файлів Windows (файли .EXE) або програм Macintosh (файли .APP), програми Java не запускаються безпосередньо операційною системою. Натомість програми Java інтерпретуються віртуальною Java машиною або JVM, яка працює на декількох платформах. Це означає, що всі програми Java є мультиплатформними та можуть працювати на різних платформах, включаючи комп`ютери Macintosh, Windows та Unix. Однак, для виконання Java програми повинен бути встановлений JVM.

Порівняно з C ++ (іншою об`єктно-орієнтованою мовою), код Java працює трохи повільніше (через JVM), але він більш портативний і має набагато кращі функції безпеки. Віртуальна машина забезпечує ізоляцію між

ненадійною програмою Java та ПК, на якому працює програмне забезпечення. Синтаксис Java схожий на C ++, але це дві зовсім різні мови програмування. Наприклад, Java не дозволяє програмістам здійснювати перевантаження операторів, як це робить C ++. Крім того, Java - це динамічна мова, де ви можете безпечно змінювати програму під час її запуску, тоді як C ++ цього не дозволяє. Це особливо важливо для мережевих додатків, які не можуть дозволити собі простоїв. Крім того, всі базові типи даних Java визначені заздалегідь і не залежать від платформи, тоді як деякі типи даних можуть змінюватися за допомогою платформи, що використовується в C або C ++ (наприклад, тип int).

Окрім мобільності, ще однією з ключових переваг Java є набір функцій захисту, які захищають ПК, що працює під управлінням програми Java, не лише від проблем, викликаних помилковим кодом, але й від шкідливих програм (наприклад, вірусів). Java також використовує автоматичний Garbage Collector (“збирач сміття”) для управління пам`яттю в життєвому циклі об`єкта. Програміст визначає, коли об`єкти створюються, а віртуальна Java машина відповідає за звільнення пам`яті, коли об`єкти більше не використовуються. Як тільки не залишилося жодних посилань на об`єкт, недосяжна пам’ять помічається особливою міткою та автоматично звільняється Garbage Collector. Це дозволяє більш швидко писати код тому що програміст не повинен піклуватися що створені ним об’єкти знищуються. Наприклад у мові програмування C++ програміст повинен сам реалізовувати алгоритм звільнення пам’яті об’єктом для кожного створеного ним класу.

Інколи так трапляється що об’єкти посилаються один на іншого та лічильник посилань на об’єкт ніколи не є рівним нулю. У такому разі Garbage Collector починає пошук живих об'єктів з області статичної пам'яті, стека та об’єктів Old Generation (ті об'єкти, які вже вижили в результаті декількох збірок сміття). Статичні члени, об'єкти з стека і Old Generation об’єкти разом утворюють так званий GC Roots - кореневу множину об'єктів. Якщо

перевіряємий Збирачем Сміття об'єкт можна досягти з цієї кореневого множини, то він – не є сміттям і позначається як живий об'єкт. Для визначення того, що сміття, а що ні, нам потрібно пройтися по всьому дереву об'єктів в купі, для чого виконання програми потрібно зупинити. Такі зупинки в роботі Збирача Сміття називаються STW (Stop-The-World) паузою і чим вона коротша, тим краще.

Недоліком Garbage Collector є непередбачуваність початку його роботи. Зазвичай віртуальна машина Java починає процес знищення об’єктів у пам’яті лише коли залишилось дуже мало вільного місця у ній. Сам процес також потребує значних ресурсів та уповільнює роботу програми. Але на практиці дуже рідко виникають ситуації коли програма створює багато об’єктів великого розміру и одразу ж знищує посилання на них.

1.4.2 Застосування графічної бібліотеки Swing

Для створення інтерфейсу програми використовується фреймворк Swing. Java Swing - це частина класів Java Foundation (JFC), яка була розроблена для того, щоб забезпечити масштабну корпоративну розробку Java-додатків. Java Swing - це набір API, що забезпечує графічний інтерфейс користувача (GUI) для програм Java. Java Swing також відомий як інструментарій віджетів Java GUI.

Java Swing або Swing був розроблений на основі більш ранніх API під назвою Abstract Windows Toolkit (AWT). Swing забезпечує більш багаті та досконалі компоненти GUI, ніж AWT. Компоненти GUI варіюються від простого текстового напису до складного дерева або таблиці. Окрім емуляції зовнішнього вигляду різних платформ, Swing також надає можливість писати елементи інтерфейсу, які не будуть залежати від платформи на якій відбувається їх робота.

Для створення графічного інтерфейсу додатку необхідно використовувати спеціальні компоненти бібліотеки Swing, звані контейнерами вищого рівня (top level containers). Вони являють собою вікна операційної системи, в яких розміщуються компоненти користувальницького інтерфейсу. До контейнерів вищого рівня відносяться вікна JFrame і JWindow, діалогове вікно JDialog, а також аплет JApplet (який не є вікном сам по собі, але теж призначений для виведення інтерфейсу в браузері, який запускає цей аплет). Контейнери вищого рівня Swing є ресурсо-важкими компоненти і є винятком із загального правила. Всі інші компоненти Swing є ресурсо-легкі.

1.4.3 Застосування Microsoft SQL Server

Для зберігання даних було обрано Microsoft SQL Server. Microsoft SQL Server - це система управління реляційними базами даних (RDBMS), яка на своєму фундаментальному рівні зберігає дані в таблицях. Таблиці - це об`єкти бази даних, які діють як контейнери для даних, в яких дані будуть логічно організовані у форматі рядків і стовпців. Кожен рядок вважається сутністю, що описується стовпцями, що містять атрибути сутності. Наприклад, таблиця клієнтів містить один рядок для кожного клієнта, і кожен клієнт описується стовпцями таблиці, в яких зберігаються відомості про клієнта, такі як CustomerName та CustomerAddress. Рядки таблиці не мають заздалегідь визначеного порядку, так що для відображення даних у певному порядку вам потрібно буде вказати порядок повернення рядків. Таблиці можуть також використовуватися як механізм безпеки, де користувачі бази даних можуть бути надані дозволи на рівні таблиці. Перевагами MS SQL Server є:

Підтримка пристроїв постійної пам`яті (PMEM).

SQL Server 2019 надає підтримку пристроїв постійної пам’яті (PMEM). SQL Server безпосередньо отримує доступ до пристрою, минаючи стек зберігання операційної системи для файлів, розміщених на пристрої PMEM.

Покращення індексу стовпців.

SQL Server 2019 також забезпечує вдосконалення функцій індексу стовпців, таких як підтримка індексу стовпців, краще управління пам’яттю метаданих, навантаження з малою пам’яттю для стовпців та покращену продуктивність для масового завантаження в індекси стовпців.

Дані завжди зашифровані захищеними анклавами.

SQL Server 2019 представляє технологію безпечного анклаву. Захищений анклав розширює клієнтські програми, передаючи перевірку довіри даним на сторону сервера. Це захищає дані від шкідливих програм та привілейованих користувачів.

1.4.4 Застосування системи управління проектами Maven

Одним з основних компонентів системи є Maven. Maven - це інструмент управління проектом, який надає розробникам повноцінну структуру життєвого циклу білдів. Команда розробників може майже миттєво автоматизувати інфраструктуру збирання проекту, оскільки Maven використовує стандартний макет каталогів та дефолтний життєвий цикл білду. У випадку декількох команд розробників Maven може стандартизувати та узгодити процес розробки за дуже короткий час. Оскільки більшість налаштувань проекту прості та легко застосовні, Maven робить життя розробника легким під час створення звітів, перевірок, збірки та тестування проекту.

Основою Maven є POM (Project object model) файл. Це загальна модель проекту. У ній описуються такі загальні характеристики як ім'я, версія, автори та їх контактна інформація, VCS проекту і пов'язані з ним мережеві ресурси, тип проекту (наприклад бібліотека або web-модуль, в оригінальній термінології називається packaging), зв'язки з іншими проектами, плагіни, які використовуються при сбірці та описи способу їх задіяння. POM допускає три типи зв'язків з іншими модулями: залежність, включення і успадкування.

Залежність, це зв'язок який вказує, що для деяких фаз життєвого циклу модуля, потрібні деякі артефакти модулів-залежностей. Що конкретно і на якій фазі життєвого циклу потрібно визначається так званої областю дії залежності. Maven розуміє кілька зумовлених областей дії і дозволяє додавати свої. Наприклад зона дії compile каже, що бінарні збірки залежності потрібні на етапі компіляції, виконання тестів і під час виконання програми.

Включення вказує, що пов'язаний модуль є невід'ємною частиною нашого модуля. Для проходження нашим модулем деякої фази життєвого циклу, включення у нього також повинен пройти цю фазу. Класичний приклад enterprise-модуль, що включає в себе web-модуль, пакет з EJB і спільну бібліотеку. Очевидно, що його збірка вимагає складання кожного з включених модулів.

Успадкування. Такий зв'язок означає перенесення на спадкоємця частини моделі предка. Правила перенесення дещо складні, але в основному діє принцип - значення параметра моделі предка стає умовчанням для моделі нащадка.

Репозиторії артефактів є виділеними сховищами результатів збірки, влаштованими так, щоб спростити пошук потрібного артефакту (по імені, версії, типу артефакту). Вони дозволяють групі розробників користуватися результатами роботи один одного без необхідності мати копії вихідних кодів їх модулів і виконувати складання дерева залежностей з 0. Maven-сумісні репозитарії стали стандартом де факто публікації результатів різних відкритих проектів.

Потрібно зауважити, що pom-файли, що містять модель проекту, також є артефактами і можуть зберігається в репозиторіях разом з іншими. Таким

чином репозиторій є у тому числі джерелом інформації (як мінімум достатньою для дистрибуції) про збережені в ньому модулі.

1.4.5 Застосування середа для розробки Intelijj IDEA

Також для розробки було використано середу розробки (IDE) JetBrains IntelliJ IDEA. IntelliJ IDEA - це інтелектуальна Java IDE, яка забезпечує надійну комбінацію інструментів розробки. Основні особливості IntelliJ IDEA включають: інтелектуальну допомогу в кодуванні, розумну навігацію та пошук, численні перероблювання, аналіз коду, підтримку розробки застосунків для Інтернету або корпоративного сектору, unit тестування та полегшення роботи в команді. Функціонал IntelliJ IDEA постійно розширюється розробниками та користувачами за допомогою плагінів. IntelliJ IDEA пропонує підтримку Java EE, Spring / Hibernate та інших технологічних стеків. Головні її переваги: Поглиблене розуміння коду - IntelliJ IDEA аналізує введений код, шукаючи зв'язки між символами для всіх файлів проекту та мов. Використовуючи цю інформацію, він надає допомогу в кодуванні, швидку навігацію, розумний аналіз помилок і допомагає проводити рефакторинг.

Розумне завершення - Ctrl + Shift + Space дає вам список найбільш релевантних символів, застосованих у поточному контексті. Це та інші доповнення постійно вивчаються, переміщуючи членів класів та пакетів, які найчастіше використовуються, до початку списку пропозицій, щоб користувач міг швидше їх вибрати.

Статичне завершення членів класу - Дозволяє легко використовувати статичні методи або константи. Пропонує список символів, що відповідають вводу користувача та автоматично додає необхідні заяви про імпорт.

Виявлення дублікатів - знаходить повторювані фрагменти коду на льоту. Якщо користувач набирає змінну, константну систему чи метод, IntelliJ IDEA повідомить про те, що вже існує аналогічний фрагмент коду.

1.4.6 Застосування фреймворку для роботи з браузером Selenium

SELENIUM - це безкоштовна (з відкритим кодом) система автоматизованого тестування, яка використовується для перевірки веб-додатків у різних браузерах та платформах. Ви можете використовувати будь-яку з мов програмування Java, C #, Python тощо, щоб створювати сценарії тестів Selenium. Тестування, проведене за допомогою інструменту Selenium, зазвичай називають Selenium Testing.

Програмне забезпечення Selenium - це не єдиний інструмент, а набір програмного забезпечення, кожене з яких відповідає потребам певного процесу тестування. На Рисунку 1 зображено повний перелік інструментів Selenium

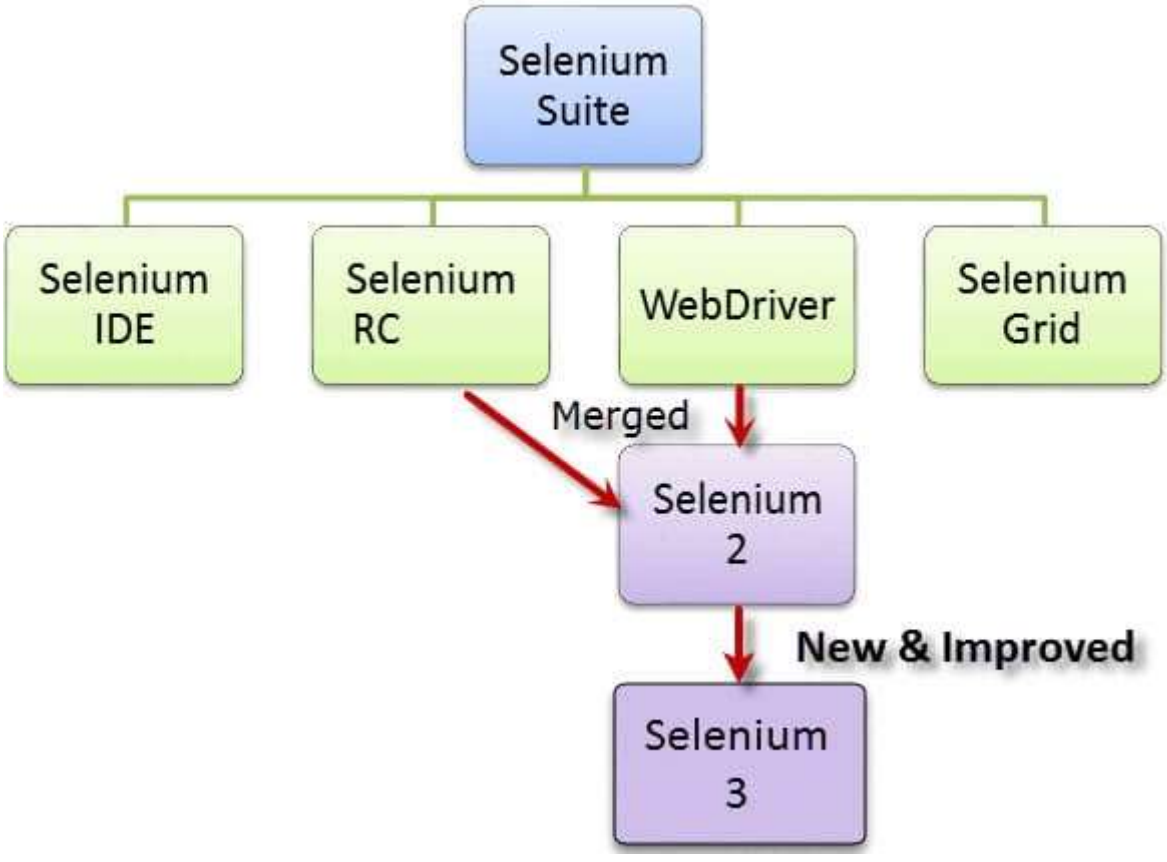


Рисунок 1 – Інструменти Selenium

Selenium був створений Джейсоном Х'юггінсом в 2004 році. Як інженер компанії ThoughtWorks він працював над веб-додатком, який вимагав частих тестувань. Зрозумівши, що повторювана ручна перевірка программного

застосунку стає все більш неефективною, він створив програму на JavaScript, яка б автоматично контролювала дії браузера. Він назвав цю програму "JavaScriptTestRunner". Побачивши потенціал у цій ідеї для автоматизації інших веб-додатків, він зробив JavaScriptRunner з відкритим кодом, який згодом був перейменований у Selenium Core.

Політика Same Origin забороняє коду JavaScript отримувати доступ до елементів із домену, який відрізняється від того, де він був запущений. Наприклад, HTML-код на сайті www.google.com використовує програму JavaScript "randomScript.js". Ця політика щодо джерела дозволить randomScript.js отримувати доступ лише до сторінок google.com, таких як google.com/mail, google.com/login або google.com/signup. Однак він не може отримати доступ до сторінок з різних сайтів, таких як yahoo.com/search або microsoft.com, оскільки вони належать до різних доменів.

Це причина, чому перед Selenium RC тестерам потрібно було встановити локальні копії як Selenium Core (програми JavaScript), так і веб-сервера, що містить веб-додаток, що тестується, для того щоб вони належали до одного домену. Через це інший інженер ThoughtWork, Пол Хаммант, вирішив створити сервер, який буде виконувати функцію проксі-сервера HTTP, щоб "обдурити" браузер, та заставити його вважати, що Selenium Core та веб-додаток, що тестується, походять з одного домену. Ця система стала називатися Selenium Remote Control або Selenium 1.

У 2006 році через те, що браузери та веб-додатки ставали все більш потужними та більш обмежуючими з такими програмами написаними на JavaScript, як Selenium Core, інженер Саймон Стюарт створив WebDriver. Це була перша міжплатформна система для тестування, яка могла контролювати браузер на рівня операційної системи.

1.5 Застосування принципів дизайну інтерфейсу десктопних застосунків

Попередження надлишку функцій. Занадто часто, щоб зробити додаток більш потужним, команди розробників додають якомога більше функцій. Це призводить до явища, відомого як надлишок функцій – постійне та надмірне додавання нових функцій у продукт. Надлишок функцій не тільки збільшує час, необхідний для створення продукту, але й робить продукт складнішим. Ідеальний інтерфейс додатку є потужним і, водночас, простим.

Функціональність ні в якому разі не можна плутати з кількістю функцій, бо вона є вищою мірою потужності. Ось чому так важливо ретельно збалансувати вибір функцій. Треба розуміти бажання цільових користувачів та створювати набір функцій, який дозволяє їм легко досягти своїх цілей. Також потрібно пам’ятати, що нема нічого поганого у видаленні всіх функції, які, ймовірно, не будуть використані.

Звичний досвід користування. Один з евристик зручності користування Якоба Нільсена говорить: "Система повинна говорити мовою користувачів зі словами, фразами та поняттями, знайомими користувачеві, а не термінами, орієнтованими на систему". Саме тому найкращий дизайн розроблюється під користувачів. Завдяки цьому не потрібно пояснювати як користуватись програмою, оскільки люди знають, як це робити з самого початку. Ось чому, перш ніж починати розробляти свій продукт, потрібно провести дослідження користувачів та аналіз конкурентів. Треба дослідивши, як люди, які представляють цільову аудиторію, використовують продукти та порівняти існуючі рішення у ніші розробляємого продукту.

Важливі елементи мають бути видимими або легко знаходжуваними. Меню або будь-які важливі дії повинні бути легко досяжними в інтерфейсі користувача. Користувачеві не слід шукати їх або покладатися на свою пам'ять, щоб знайти належний контроль інтерфейсу користувача.

Працюючи над інтерфейсом робочого столу, потрібно зменшити візуальне захаращення та розмістити елементи в потрібному місці всередині вікна. Як правило, розташування елемента має відповідати його корисності. Наприклад, користувачі очікують побачити параметри навігації та важливі дії у верхній частині вікна.

Зрозумілі піктограми. Значення піктограм у інтерфейсі повинно бути чітко зрозумілим для користувачів. Вони повинні зрозуміти значення іконки, просто подивившись на неї. Деякі дизайнери вважають, що можна використовувати підказку для значків та інших інтерактивних елементів, щоб передати значення. Але підказки - не найкраще рішення, оскільки вони збільшують тривалість взаємодії - користувачеві потрібно почекати секунду, щоб побачити значення елемента. Щоразу, коли користувач бачить значок, він повинен знати його значення, ще до того як наведе на нього курсор. Якщо є сумніви, чи виникнуть проблеми у користувачів з розумінням значення піктограми, краще замінити її на більш чітку альтернативу або використовувати мітку разом з нею.

Послідовність у дизайні для програмних застосунків. Практично будь-який посібник для мобільних пристроїв говорить, що дизайн продукту повинен бути схожим між усіма платформами. Тож у користувачів Android не повинно виникнути проблем із переходом на iOS, і навпаки. Це ж правило стосується і програм для стаціонарних компьютерів. Існують три основні платформи настільних ПК – Windows, Mac та Linux. Саме тому метою є забезпечити, щоб дизайн продукту дозволяє користувачам переходити з однієї платформи на іншу, коли це необхідно і не відчувати різниці.

Забезпечити правильну щільність інформації. Щільність інформації - це кількість інформації в одиниці екранного простору. Мобільний дизайн багато в чому полягав у розміщенні якомога більше інформації на маленькому екрані. Невеликий екран мобільних пристроїв змушує розробників розділити

інформацію на кілька екранів і розмістити на кожному обмежену кількість інформації. І хоча екрани стаціонарни комп'ютерів мають набагато більше простору, все одно буде корисним зменшіти загальну кількість кроків, необхідних для досягнення мети, надаючи більше інформації про кожен крок.

Треба мінімізувати навантаження пам'яті користувача, зробивши видимі об’єкти, дії та параметри. Особливо важливо зосередитись на створенні гарної навички навігації – десктопні програми можуть підтримувати фіксовану панель навігації, наприклад.

Дизайн у вигляді сітки. У порівнянні з мобільними пристроями, які в основному підтримують макет з одним стовпцем, робочий стіл з великим екраном дозволяє використовувати сітку для організації вмісту. Будь-який вміст на робочому столі може відображатися у форматі багатьох стовпців, а ширина стовпців має визначається за допомогою відсотків, а не фіксованих значень, щоб вміст міг адаптуватися до будь-якого розміру екрана.

Проектування для забезпечення тривалих сеансів користування. Користувачі віддають перевагу стаціонарним комп’ютерам саме для більш довгих завдань, виконання яких вимагає більше часу. Коли виникають довші завдання, особливо важливо враховувати комфорт користувача. Треба переконайтися, що користувачі відчувають себе комфортно під час роботи з продуктом. Наприклад, тривалі сеанси читання не повинні викликати перенапруження очей, або довгі сеанси взаємодій не повинні спричиняти фізичний біль у тілі користувача.

1.6 Методи захисту конфіденційності медичних даних

Найкращі практики щодо кібербезпеки систем пов’язаних з охороною здоров'я мають на меті йти в ногу з розвитком кіберзагроз, вирішуючи загрози конфіденційності та захисту даних у кінцевих точках та у хмарних обчисленнях, а також захист даних під час транзиту, спокою та в користуванні. Для цього потрібен багатогранний підхід до безпеки.

Підвищення комп’ютерної освідченості медичних працівників. Людський елемент залишається однією з найбільших загроз безпеці у всіх галузях, особливо в галузі охорони здоров'я. Прості людські помилки чи недбалість можуть призвести до катастрофічних та дорогих наслідків для медичних організацій. Навчання з питань безпеки з боку охорони здоров'я забезпечує працівників охорони здоров'я знаннями, необхідними для прийняття розумних рішень та дотриманню відповідної обережності при обробці даних пацієнтів.

Обмежений доступ до даних та програмних застосунків. Реалізація контролю доступу посилює захист конфіденційних медичних даних, дозволяючи доступ до інформації про пацієнтів та певних програм лише тим користувачам, яким він потрібен для виконання своїх завдань. Обмеження доступу вимагають аутентифікацію користувача, гарантуючи, що тільки авторизовані користувачі мають доступ до захищених даних. Багатофакторна автентифікація - це рекомендований підхід, який вимагає від користувачів підтвердження того, що вони насправді є особою, уповноваженою отримувати доступ до певних даних та програм, використовуючи два чи більше методів перевірки, включаючи:

- Інформацію, котра відома лише користувачеві, наприклад, пароль або PIN-код.
- Щось, що матиме лише авторизований користувач, наприклад, карта чи ключ.

- Щось унікальне для авторизованого користувача, наприклад, біометричні дані (розпізнавання обличчя, відбитки пальців, сканування очей)

Контроль використання даних. Захисні засоби контролю даних виходять за рамки звичайного контролю доступу та моніторингу, щоб гарантувати, що ризиковану або зловмисну активність можна було позначити та / або заблокувати в режимі реального часу. Організації охорони здоров'я можуть використовувати засоби контролю даних для блокування конкретних дій, пов'язаних із конфіденційними даними, такими як завантаження в Інтернет, несанкціоноване надсилання електронної пошти, копіювання на зовнішні диски або друк. Відкриття та класифікація даних відіграють важливу роль у цьому процесі, забезпечуючи, щоб конфіденційні дані можна було ідентифікувати та позначити для отримання належного рівня захисту.

Використання логування та моніторингу активності. Логування всіх даних про доступ та використання медичних даних також має важливе значення, тому що дозволяє керівництву лікарні контролювати, які користувачі отримують доступ до якої інформації, програм та інших ресурсів, коли саме та з яких пристроїв чи місцеположень. Ці записи є цінними для аудиторських цілей, допомагаючи організаціям визначити проблемні сфери та посилити захисні заходи, коли це необхідно. Якщо трапляється інцидент, аудит може допомогти організаціям визначити точні точки входу, визначити причину та оцінити збитки.

Шифрування даних у стані покою та під час переміщення. Шифрування - один із найкорисніших методів захисту даних для організацій охорони здоров'я. Зашифровуючи дані під час транзиту та в стані спокої, медичні працівники ускладнюють (в ідеалі унеможливлюють) зловмисникам розшифровку інформації про пацієнтів, навіть якщо вони отримують доступ до даних. HIPAA пропонує рекомендації, але смедичні організації не вимагають

впровадження заходів шифрування даних; натомість це правило залишає за медичними працівниками право визначати, які методи шифрування та інші заходи необхідні, враховуючи робочий процес організації та інші потреби.

Забезпечення захисту мобільних пристроїв. Все частіше медичні працівники використовують мобільні пристрої під час ведення бізнесу, будь то лікар, який використовує смартфон для доступу до інформації, щоб допомогти собі у лікувані пацієнта чи адміністративного працівника, який обробляє страхові претензії. Захист мобільних пристроїв сам по собі передбачає безліч заходів безпеки, включаючи:

- Керування всіма пристроями, налаштуваннями та конфігураціями
- Забезпечення використання надійних паролів
- Увімкнення можливості віддалено стирати та заблокувати втрачені чи вкрадені пристрої
- Шифрування даних програми
- Моніторинг облікових записів електронної пошти та вкладень, щоб запобігти зараженню зловмисним програмним забезпеченням або несанкціонованому пошуку даних
- Навчання користувачів передовим практикам безпеки мобільних пристроїв
- Реалізація вказівок та процедур, щоб гарантувати встановлення лише програм, що відповідають заздалегідь визначеним критеріям або є попередньо перевіреними.
- Вимога від користувачів постійного оновлення своїх пристроїв за допомогою останніх оновлень операційної системи та додатків.
- Вимога встановлення програмного забезпечення для мобільних пристроїв, таких як спецальні програмні рішення для управління мобільними пристроями.

Проведення регулярної оцінки ризиків. Незважаючи на те, що проведення аудиту допомагає виявити причину та інші цінні деталі інциденту після його виникнення, не менш важливою є і активна профілактика. Проведення регулярних оцінок ризику може виявити вразливі місця або слабкі місця в безпеці організації охорони здоров'я, недоліки в освіті працівників, недоліки в безпеці постачальників послуг та бізнес-партнерів. Періодично оцінюючи ризик в межах організації охорони здоров’я для активної ідентифікації та зменшення потенційних ризиків, медичні працівники та їхні бізнес-партнери можуть краще уникати потенціальних порушень даних та багатьох інших згубних наслідків порушення даних, від пошкодження репутації до штрафних санкцій регуляторних органів.

2 КОНЦЕПЦІЯ СИСТЕМИ ДЛЯ ОБРОБКИ ДАНИХ ПАЦІЄНТІВ

2.1 Мета та задачі системи

Метою створенню цієї системи є полегшення виконання службових обов'язків персоналом лікарні та підвищення ефективності роботи лікарні взагалі. Щоб до досягти цієї цілі ми маємо забезпечити надійне зберігання облікових даних кожного пацієнта. Дані не повинні бути втрачені або пошкоджені. Повинен бути створений механізм який зменшить ризик помилки зі сторони персоналу лікарні та унеможливить доступ до даних пацієнта третіх осіб, які не мають на то права.

Ще однією важливою особливістю системи є сумісність з UDRG-System.com (державний сайт для розрахунку вартості медичних послуг та вагових коефіцієнтів в закладах охорони здоров'я України). На даний момент заповнення анкет про госпіталізації пацієнтів на цьому сайті є одним з обов'язків персоналу лікарень. Цей процес займає значний проміжок часу та ніяк не допомагає у наданні медичних послуг пацієнту на рівні лікарні, адже відправленні дані використовуються лише у звітах та статистиці.

Саме тому, однією з основних функцій розробленої системи є надання можливості авторизованим користувачам можливість перегляду попередньо внесених даних, пошук по ключовим словам, представлення знайдених результатів у зручному вигляді та опція збереження знайдених даних у форматі PDF.

Ще однією важливою функцією є можливість заповнити попередньо збереженими даними анкету у UDRG-System.com. Це дає змогу лікарям не залежати від робочого стану сайту та заповнювати анкети саме тоді, коли вони вважають за потрібне це роботи. У разі технічних робіт на сервері, користувач може вносити дані у локальну базу даних не використовуючи мережу Інтернет.

Створений програмний застосунок повинен мати зручний для роботи інтерфейс, графічні елементи та шрифти повинні мати оптимальний розмір щоб запобігти втомі користувача під час вводу великої кількості даних.

2.2 Структура системи

Розроблена система має складатися з декількох частин – модулю для роботи з базою даних, графічного інтерфейсу для внесення даних, графічного інтерфейсу для здійснення пошуку та відображення даних, модулю для автоматичного заповнення анкети на сайті UDRG-System.com та авторизаційного модулю.

2.2.1 Модуль для роботи з базою даних

Модуль для роботи з базою даних повинен виконувати три базові функції. По-перше він повинен допомагати обмежувати доступ до важливої інформації людям, які не мають дозволу на роботу з нею. Будь-яка взаємодія з локальною базою даних має відбуватися лише через нього. Це дозволить підняти безпеку збереження даних від пошкодження або крадіжки. Також мають існувати різні види дозволів на роботу з даними. Наприклад, повинна існувати можливість зробити так що лише головний лікар має право видаляти чи змінювати збережені дані.

По-друге цей модуль повинен забезпечувати безвідмовне збереження даних введених користувачем. Окрім даних які вводяться на сайті UDRG-System.com повинні також зберігатися службові дані, наприклад час створення запису та відмітка про успішність внесення даних на сайті. Час створення запису в майбутньому дозволить нам виявляти причини змін збережених даних та автора цих змін у разі виникнення помилок. Відмітка про успіх збереження анкети на сайті потрібна у разі технічних робіт або порушень у роботі сайту та відправлення збережених даних пізніше. Ще одним важливим фактором є збереження даних у відповідному форматі. Усі поля які мають необмежену

довжину повинні відповідно зберігатися, а дати зберігатися саме у форматі дати. Це дозволить запобігти втраті даних.

По-третє модуль для роботи з базою даних має вміти швидко повертати попередньо збережені дані по ключовому слову. Це буде використовуватись під час перегляду та фільтрування анкет госпіталізацій пацієнтів та виводу результатів на екран. Пошук має відбуватися не лише за точним значення, але й за частковим значенням. Наприклад якщо користувач захоче знайти всі госпіталізації пацієнтів, ім’я яких має починається або має у собі літеру ‘А’, він повинен мати змогу це зробити.

2.2.2 Графічний інтерфейс для внесення даних

Графічний інтерфейс для внесення даних повинен відтворювати інтерфейс для заповнення анкет на сайті UDRG-System.com для усунення необхідності звикання користувачів за запобігання зниженню ефективності через це їх роботи. У той самий час він повинен бути більш комфортним для роботи та, за можливості, заповнювати певні поля анкети автоматично. Також він має бути масштабованим і коректно відображатися на моніторах з різною діагоналлю екрану. Шрифт, обраний для інтерфейсу має бути великого розміру та розбірливим. Поля, які мають обмежену кількість значень але їх дуже багато для того щоб відобразити їх усі у одному вікні – мають підтримувати пошук по значенню, навіть якщо це значення є неповним. Інтерфейс також має мати перевірку на заповнення обов’язкових для анкети полів та виведення на екран помилок у разі відсутності значень. Де це можливо, введені значення повинні перевірятися на їх коректність.

Також требу звернути увагу на особливі поля «Додаткові діагнози» та «Процедури» які є на сайті UDRG-System.com. Вони можуть динамічно додаватися та видалятися під час заповнення користувачем анкети і це має бути відображено у інтерфейсі програми.

2.2.3 Модуль автоматичного заповнення анкет

Дані, введені користувачем у графічному інтерфейсі внесення даних повинні бути використані для заповнення анкети на сайті. Модуль повинен вміти працювати з різними елементами веб-сторінки – такими як input та select. Однією з особливостей сайту UDRG-System.com є динімачно заповнювані елементи select. Під час завантаження сторінки вони не мають ніяких даних взагалі (Рисунок 2).

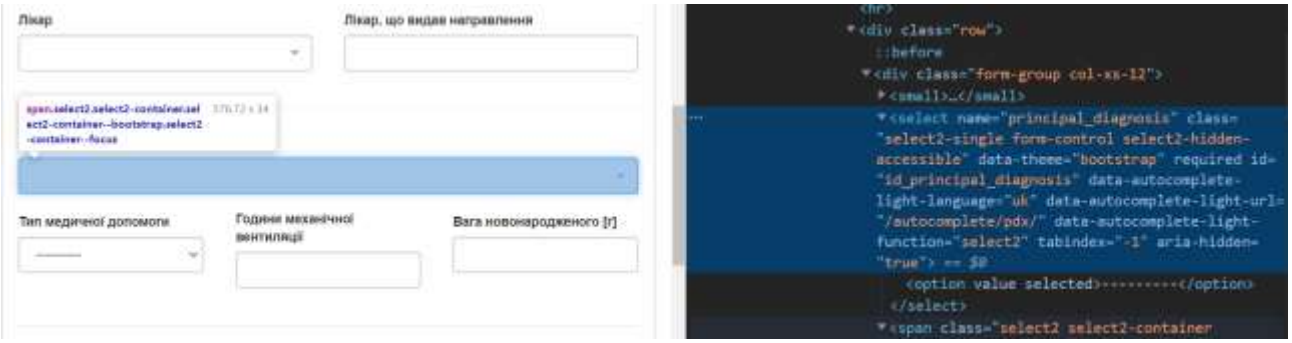


Рисунок 2 – Динамічний select одразу після завантаження

Дані починають завантажуватися лише тоді, коли користувач вводить у поле дані або відкриває випадаюче меню елемента. У разі якщо користувач ввів значення – через API запит завантажуються перші двадцять або менше можливих значень які містять у собі значення введене користувачем. Ці значення потім представляються у вигляді таблиці (Рисунок 3).

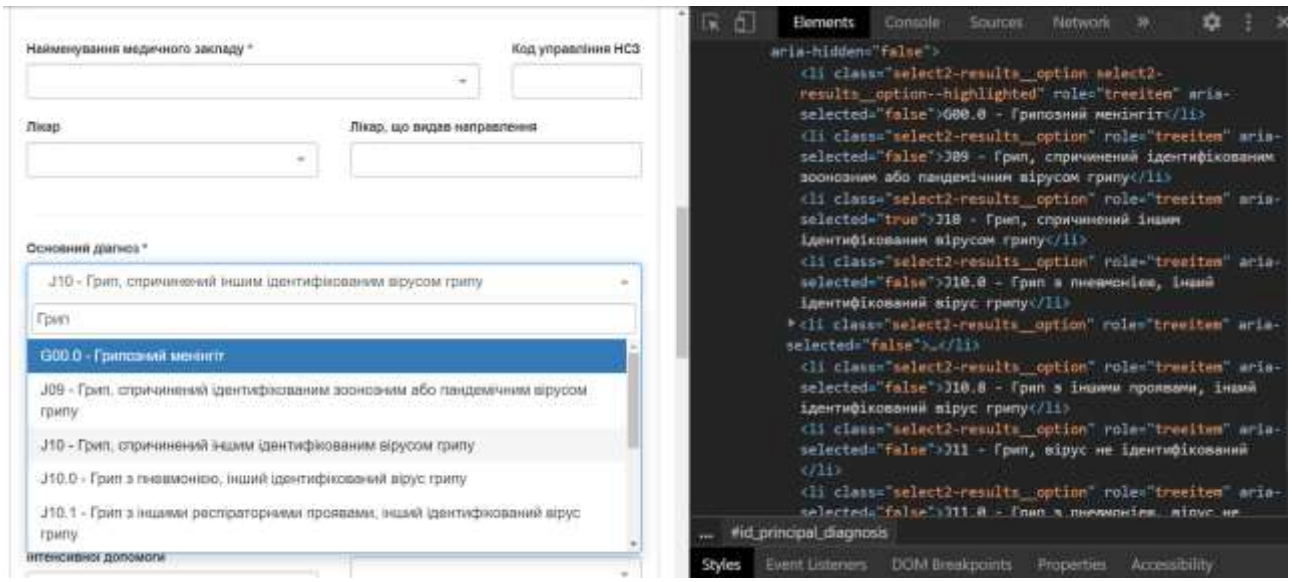


Рисунок 3 – Випадаюче меню з завантаженними значеннями

У разі якщо користувач не вводив у полі ніяких даних та одразу відкрив випадаюче віконце, API запит повертає перші двадцять можливих значень взятих з бази даних. У обох випадках якщо загальна кількість можливих значень перевищує кількість завантаженних можливих значень, то коли юзер досягне останнього можливого значення у списку буде здійсненний ще один API запит та підвантажені наступні двадцять можливих значень.

Коли користувач обирає одне з цих значень воно автоматично додається до елементу select (Рисунок 4).

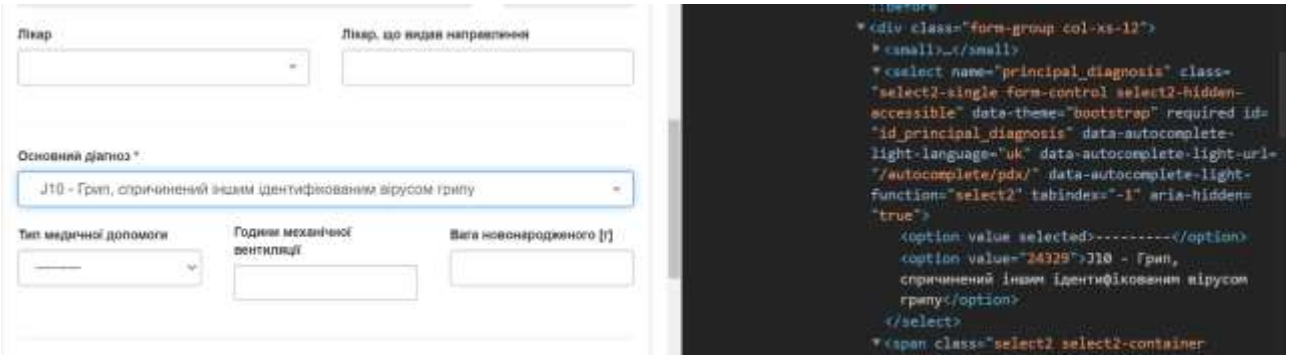


Рисунок 4 – Структура елементу select після обрання значення

Сам елемент не дозволяє вводити значення користувачу самостійно, він має обрати потрібне з випадаючого меню. Більш того, елемент обраний для демонстрації («Основний діагноз») є обовязковим для заповнення і анкету неможливо буде надіслати не заповнивши його.

Ще розроблений модуль має вміти не лише заповнювати поля «Додаткові діагнози» та «Процедури» їх кількість є змінюваною та модуль повинен додавати нові поля без допомоги користувача. Приклад полей можно побачити на Рисунку 5.

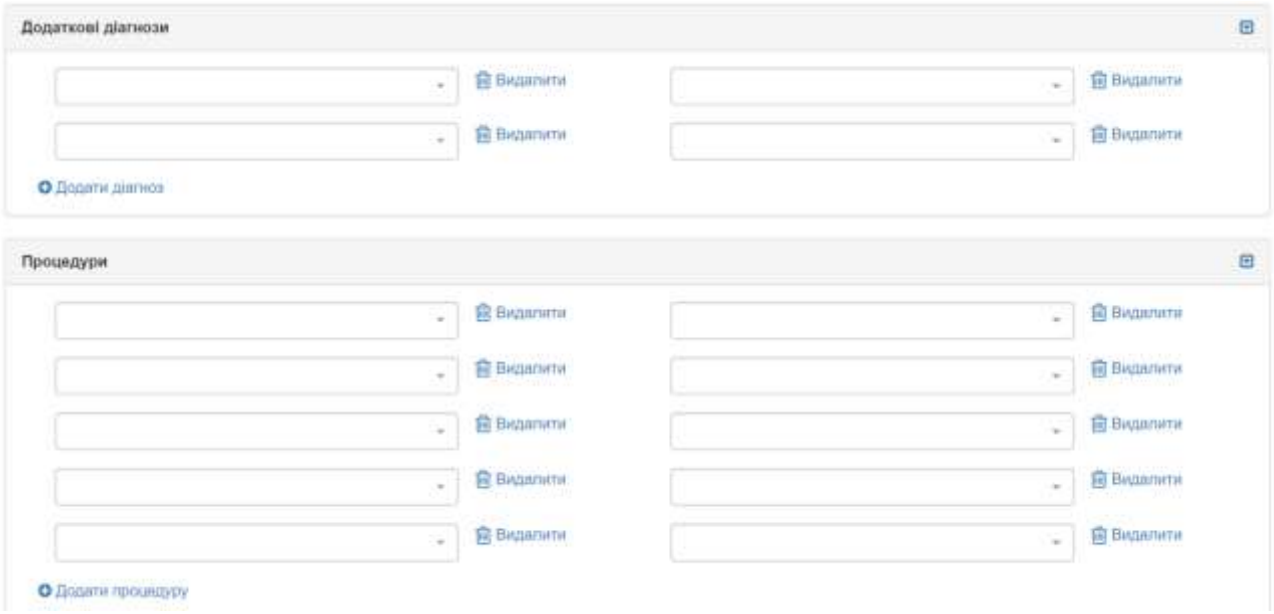


Рисунок 5 – Поля для додаткових дігнозів та процедур

2.2.4 Авторизаційний модуль

Мета цього модулю – забезпечити безпеку вводу, перегляду та зміни збереженних даних. Він повинен використовувати модуль для роботи з базою даних для перевірки облікових даних користувача під час запуску програми та обмежувати доступ відповідно до статусу ідентифікованого користувача у системі. Він також має мати графічний інтерфейс та можливість виводу повідомлень о помилці у разі некоректного вводу даних зі сторони користувача.

2.3 Обгрунтування використання фреймворків та бібліотек

2.3.1 Використання бібліотеки Swing

Перш за все, слід розуміти що під час написання інтерфейсу для Java застосунка, розробник має обрати фреймворк, у якому він буде його створювати. Тут є два найбільш поширені варіанти – Swing та Java FX.

У цьому проекті було вирішено застосувати Бібліотеку Swing через такі її переваги як портативність, відсутність додаткових залежностей та розширюваність.

Одною з найбільших переваг Swing є те, що майже на всіх операційних системах вона виглядає однаково. Це дозволяє зменшити кількість часу необхідну на разробку під декілька платформ. Описавши інтерфейс один раз, розробник може бути впевненним у тому, що на іншій платформі він не буде значно відрізнятися. Це дає ще одну перевагу – малу залежність від системних графічних елементів. Через те, що системні елементи Windows не використуваються у роботі програми, нема ніяких проблем з запуском програми на платформі MacOS.

У той же самий час, JavaFX є дуже потужним графічним інструментом і це створює певні проблеми. Для того щоб виконувати складні графічні задачі, JavaFX використовує багато нативних (написаних не на Java) бібліотек. На жаль, стабільна робота цих бібліотек у Linux-подібних системах не гарантується.

Іншою перевагою Swing є те, що вона входить у набір бібліотек Java і її не потрібно окремо завантажувати для кожного користувача. Достатньо лише наявності Java Virtual Machine на комп’ютері користувача. Для роботи JavaFX на компьютері користувача потрібно встановити JavaFX SDK або використовувати відповідні залежності Maven.

Останньою важливою перевагою Swing є можливість легко розширювати графічні елементи, майже повністю змінюючи їх зовнішній вигляд. Наприклад інтерфейс дуже популярної IDE під назвою Intelij IDEA повністю написан на Swing.

2.3.2 Використання фреймворку Selenium

Державний сайт UDRG-System.com дозволяє лише два варіанта вводу даних про госпіталізацію пацієнта. Це може бути здійснено через API виклик або через заповнення анкети автоматично, використовуючи фреймворк Selenium для імітації роботи користувача.

На перший погляд API виклик здається більш привабливим. Усі дані передаються за один раз, якщо цей процес іде не так, його дуже легко повторити. На жаль у цього варіанта є великі недоліки. У разі змін протоколу API, навіть самих незначних, цей механізм просто перестає працювати. Посилання API викликів також потребує реверс-інженерії роботи сайту, для того щоб від його імені робити виклики. Іншою проблемою є неможливість контролю над передачею даних зі сторони користувача. У разі якщо API виклик буде передавати пошкоджену інформацію, користувач не має ніякої змоги це помітити адже весь процес відбувається у фоновому режимі. І навіть якщо цей процес створював ретельний опис усіх своїх дій, для технічно непідготовленої людини це все одно не принесло би ніякої користі.

Отже було вирішено зупинитися на фреймворку Selenium. Користувач має більше можливостей контролювати процес заповнення анкети. Більш того, анкета не буде відправлена автоматично до тих пір, поки користувач не підтвердить правильність її заповнення. У разі змін у дизайні сайту, наприклад виникненні нових полів, а бо усуненні старих, скрипт для Selenium все одно зможе працювати з тими полями які є на веб-сторінці та локатори котрих йому знайомі.

3 РЕАЛІЗАЦІЯ МОДУЛЕЙ СИСТЕМИ

3.1 Принципи роботи системи

Перед початком роботи користувач має авторизуватися. Модуль авторизації є першим графічним елементом який зустрічає користувач. Без авторизації подальша робота є неможливою. Якщо логін та пароль є вірними користувач отримує доступ до інших функцій програми – насамперед інтерфейсу вводу даних про госпіталізацію пацієнта. Після її заповнення, дані зберігаються у локальну базу даних, після чого викликається WebApi модуль для заповнення анкети на сайті UDRG-System.com. Діаграму послідовностей зображено на Рисунку 6.

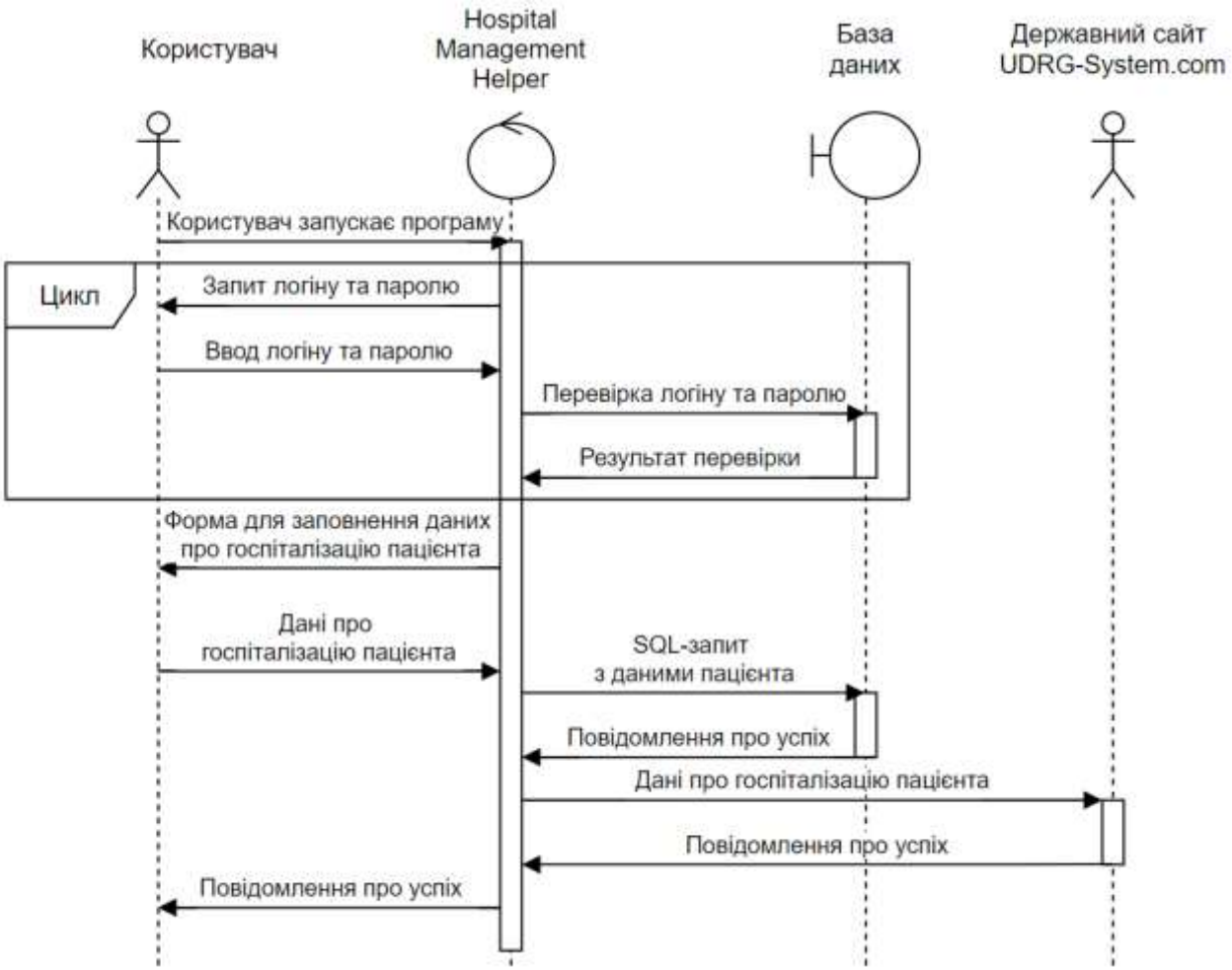


Рисунок 6 – Діграма послідовностей

Після заповнення анкети на сайті UDRG-System.com користувач може вийти з програми, заповнити нову анкету чи переглядати попередньо внесені анкети. На Рисунку 7 зображено діаграму прецедентів.

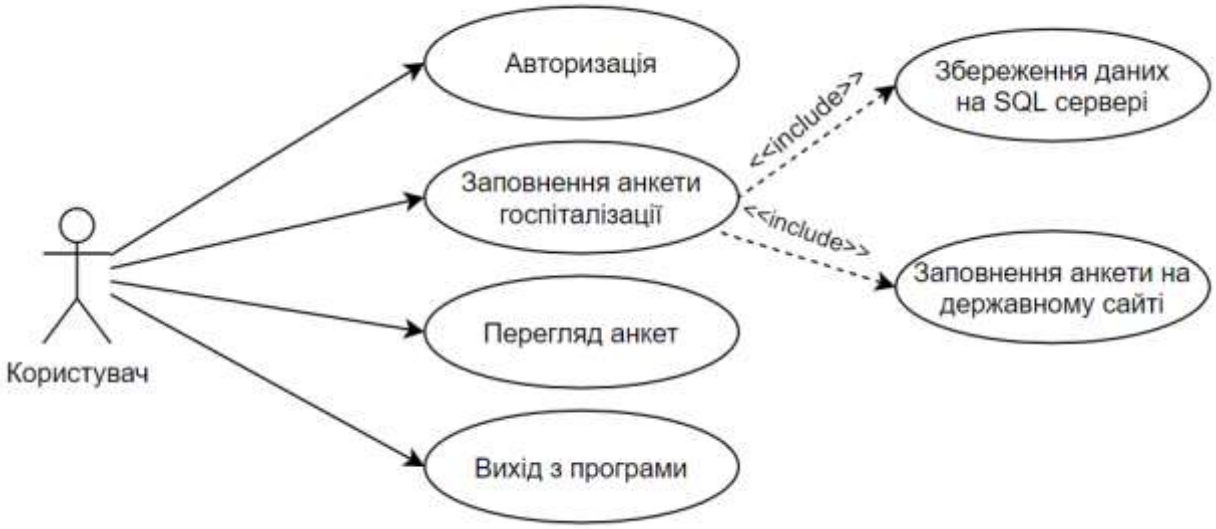


Рисунок 7 – Діаграма прецедентів.

3.2 Авторизаційний модуль

Так як код мови програмування Java є відкритим та легко декомпілюємим, було прийнято рішення перевіряти та обробляти облікові дані користувача за допомогою SQL серверу. Програмний застосунок відправляє дані на первірку серверу та чекає на відповідь, у разі виникнення проблем він виводить повідомлення про помилку (Рисунок 8).

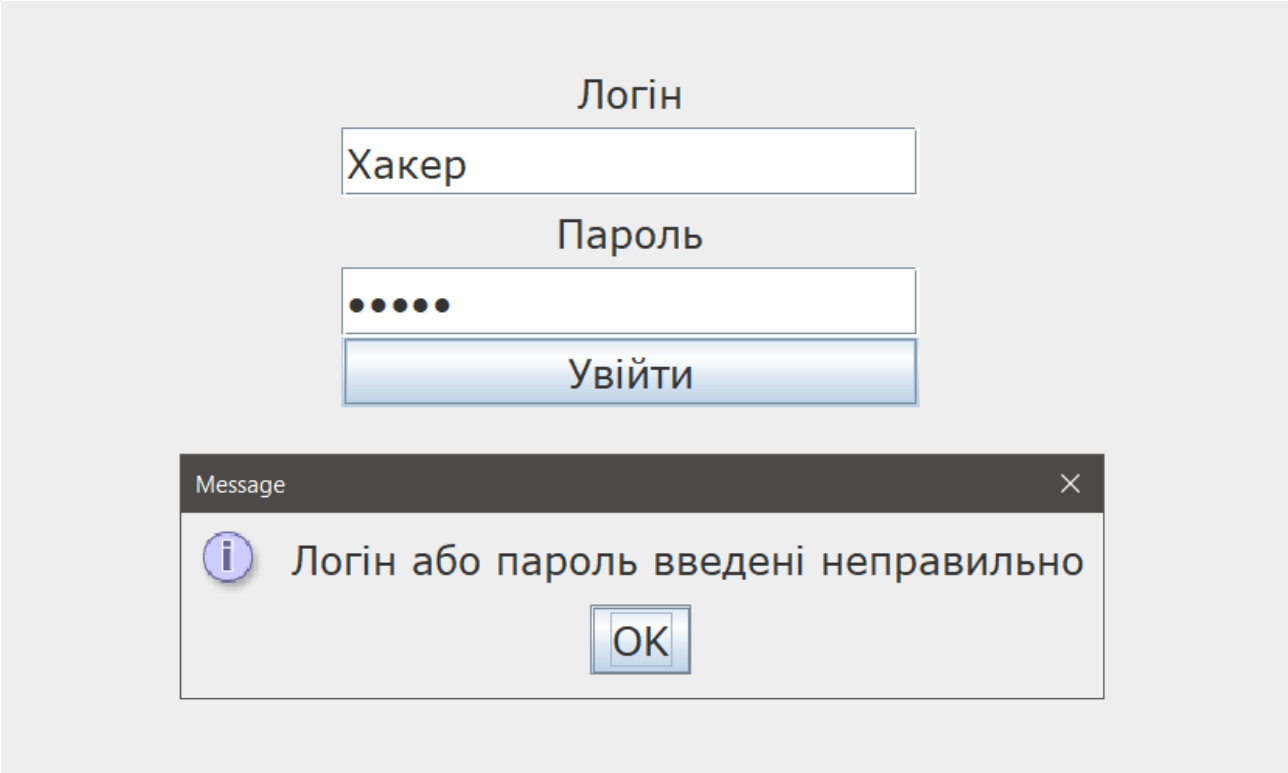


Рисунок 8 – Авторизаційне вікно та повідомлення про помилку

У разі якщо SQL сервер підтвердив коректність авторизаційних даних, навігаційний елемент розблоковується та з’являється на екрані.

3.3 Особливі типи даних

Для збереження даних так ефективної роботи з ними було створено новий тип даних Case. Він поєднує у собі усі дані які потрібні для заповнення анкет на сайті UDRG-System.com та службові дані, необхідні для покращення роботи системи. На Рисунку 9 зображена його скорочена діграма класів.

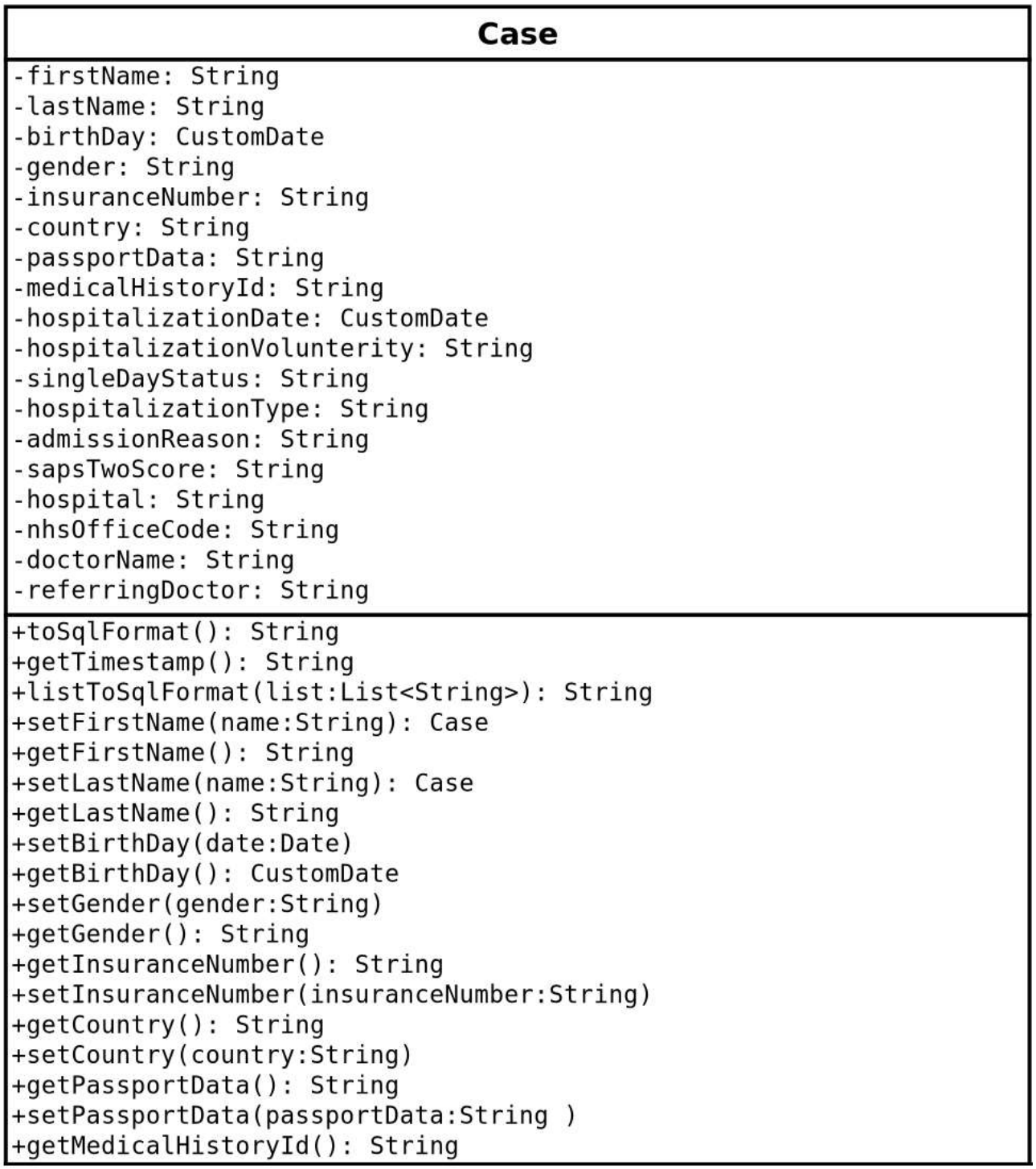


Рисунок 9 – Діграма класів нового типу даних Case

Окремо слід відзначити що цей новий тип даних має метод який повертає усі заповненні дані у вигляді об’єкту String заповненного у SQL форматі. Це зроблено для полегшення роботи по внесенню нових даних до SQL сервера. Також він має метод, який дозволяє представити свої поля, які містять списки з

необмеженою кількістю значень, у вигляді SQL запиту. Це зроблено для полів додаткових дігнозів та процедур, які можуть бути внесені у будь-якій кількості.

Через те, що поля, які містять дату, для сайту UDRG-System.com та SQL серверу відрізняються, було створено особливий тип даних CustomDate, який наслідує класу java.util.Date. Головною його особливістю є два методи для представлення дати у форматі Web та SQL відповідно. Формат дати для веб-сторінки є типовим форматом, прийнятим в Україні, а саме день-місяць-рік. У той час як SQL-севера потребує англійський формат дати, а саме рік-місяць-день. Діаграма класів нового типу даних зображена на Рисунку 10.

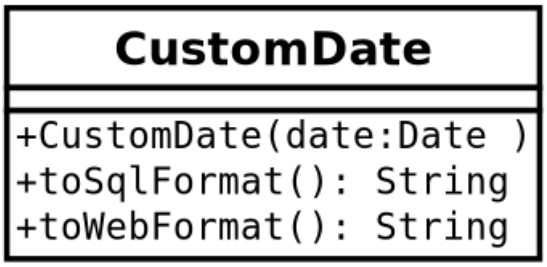
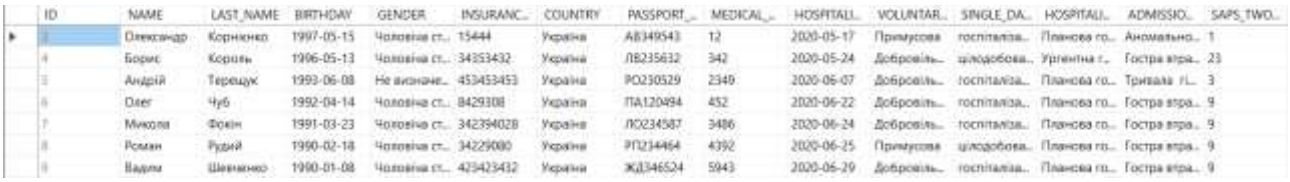


Рисунок 10 – Спеціальний тип для відображення дати у різних форматах

Також цей клас має ще один метод службового характеру, який дозволяє повернути поточну дату у форматі SQL серверу з точністю до секунди. Він використовується під час запису у базу даних.

3.4 База даних на основі Microsoft SQL Server

Для створення бази даних було використано Microsoft SQL Server. Основними його перевагами є безкоштовність обмеженої версії Miscrosoft SQL Express, підтримка мови програмування Java та операційної системи Windows. У даному випадку використано бібліотеку com.microsoft.sqlserver а саме JDBC Driver для обміну інформацією між базою даних та програмою. Приклад збереження даних у MS SQL Server сображено на Рисунку 11.



ID	NAME	LAST_NAME	BIRTHDAY	GENDER	INSURANC...	COUNTRY	PASSPORT_...	MEDICAL_...	HOSPITALI...	VOLUNTAR...	SINGLE_DA...	HOSPITALI...	ADMISSIO...	SAPS_TWO...
3	Олександр	Корніенко	1997-05-15	Чоловіча ст...	15444	Україна	АВ349543	12	2020-05-17	Примусова	госпіталіза...	Планова го...	Аномально...	1
4	Борис	Король	1996-05-13	Чоловіча ст...	34353432	Україна	ЛВ235632	342	2020-05-24	Добровіль...	цілодобова...	Ургентна г...	Гостра втра...	23
5	Андрій	Терещук	1993-06-08	Не визначе...	453453453	Україна	РО230529	2349	2020-06-07	Добровіль...	госпіталіза...	Планова го...	Тривала гі...	3
6	Олег	Чуб	1992-04-14	Чоловіча ст...	8429308	Україна	ПА120494	452	2020-06-22	Добровіль...	госпіталіза...	Планова го...	Гостра втра...	9
7	Микола	Фокін	1991-03-23	Чоловіча ст...	342394028	Україна	ЛО234587	3486	2020-06-24	Добровіль...	госпіталіза...	Планова го...	Гостра втра...	9
8	Роман	Рудий	1990-02-18	Чоловіча ст...	34229080	Україна	РТ234464	4392	2020-06-25	Примусова	цілодобова...	Планова го...	Гостра втра...	9
9	Вадим	Шевченко	1990-01-08	Чоловіча ст...	423423432	Україна	ЖД346524	5943	2020-06-29	Добровіль...	госпіталіза...	Планова го...	Гостра втра...	9

Рисунок 11 – Приклад збереження даних у БД

3.5 Модуль SqlApi та Properties

На цьому модулі лежать всі функції обміну інформацією з SQL сервером. Він має функцію перевірки авторизаційних даних користувача, функцію внесення даних до SQL серверу та отримання даних знайдених за певним критерієм. Функція для внесення даних до SQL серверу приймає об’єкт типу Case з якого вона бере дані у формі SQL запиту.

Клас properties виконує подібну до SqlApi роль, але працює з Json файлами. Він створений для роботи з певними значеннями, які було не дуже зручно записувати до SQL серверу насамперед через можливу необхідність внесення швидких змін через, наприклад, зміну можливих значень на сайті UDRG-System.com. Таку схему було обрано для даних типу діагнози, процедури, відділень лікарні та лікарень до яких переводиться хворий. Список діагнозів налічує майже двадцять тисяч захворювань, а список процедур трохи більше ніж шість тисяч назв.

3.6 Графічний інтерфейс заповнення анкет

Цей модуль використовується для внесення даних користувачем. Ряди графічних елементів у ньому об’єднані у окремі панелі, які наслідують класу MyCasePanel та наслідють від нього абстрктний метод fillCase, який приймає та повертає об’єкт типу Case. Це зроблено для уніфікованого інтерфейсу сбору даних розміщених на панелі, достатньо передати пустий об’єкт Case та імплементуюций цей інтерфейс клас запише внесені користувачем значення у відповідні поля. Також від MyCasePanel наслідується метод getPrefferedSize який дозволяю виставити висоту усіх панелей як одну шосту від висоти екрана.

Загальний вигляд інтерфейсу зображено на Рисунку 12.

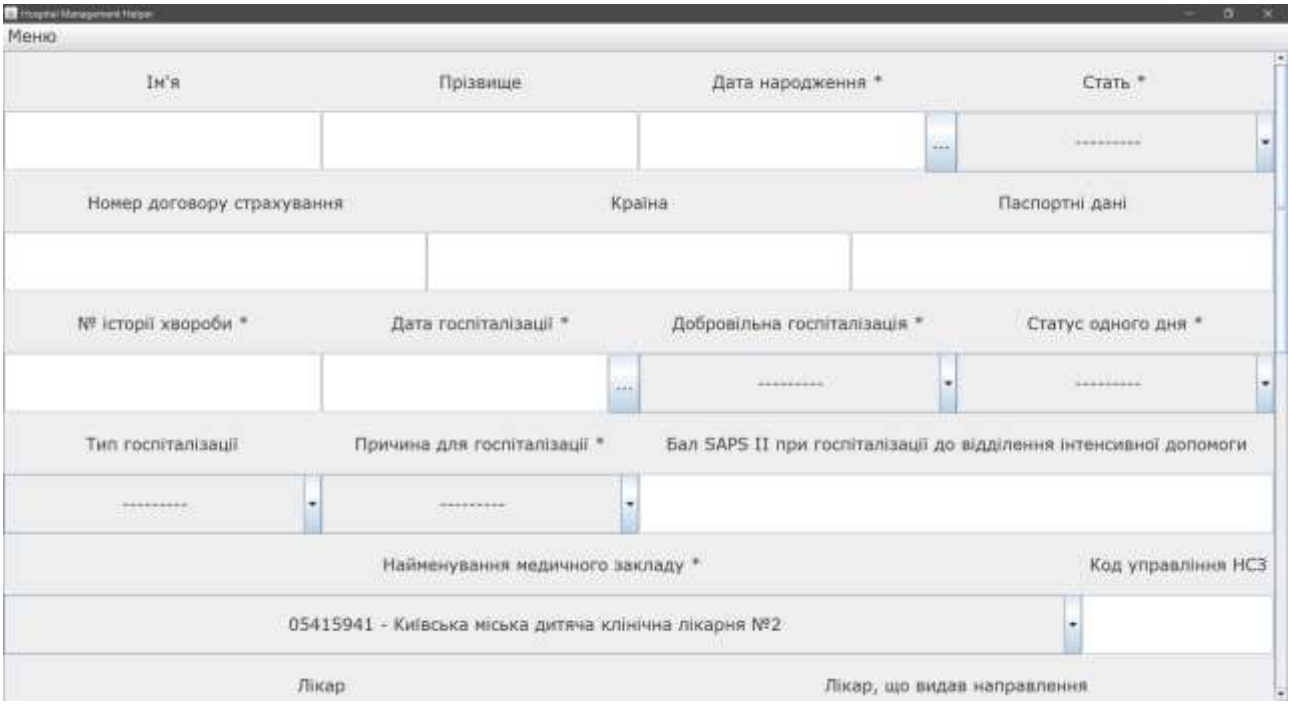


Рисунок 12 – Загальний вигляд інтерфейсу для заповнення анкет

Для вводу дат взято графічний календар org.jdatepicker та додані до нього деякі особливості. Наприклад змінено LayoutManager для його внутрішніх елементів з метою покращити зовнішній вигляд. Також під час відкриття випадаючого меню календаря його ширина ніколи не перевищує ширину батьківського об’єкту по вона вираховується динамічно. Приклад роботи цього елементу можна побачити на Рисунку 13.

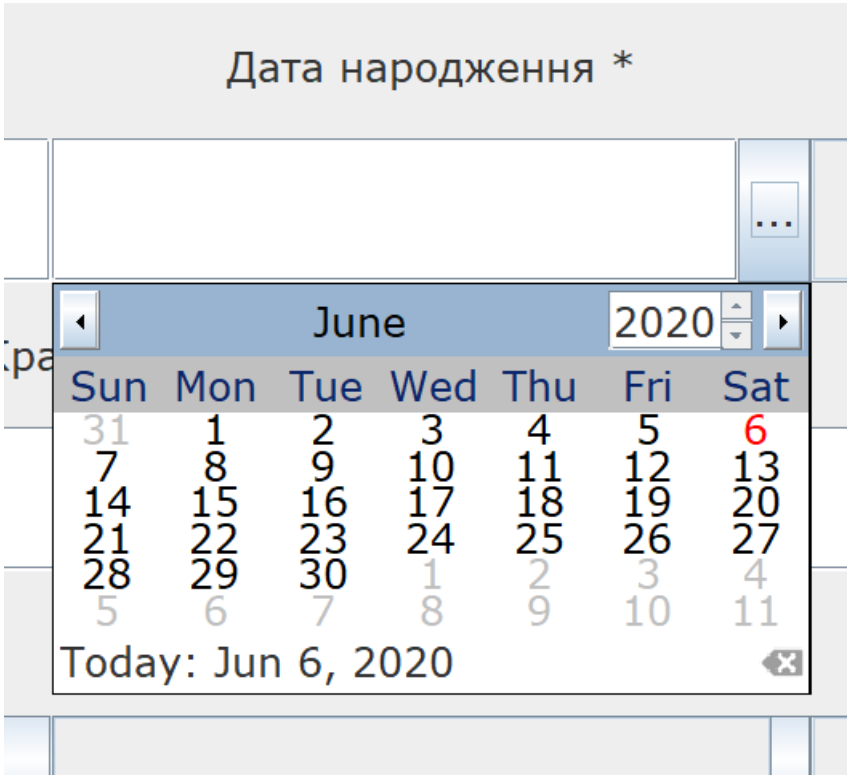


Рисунок 13 – Календар MyJDatePicker

Цей елемент автоматично обирає мову для відображення календаря таку ж, яка використовується користувачем на комп'ютері, у наведеному випадку це англійська локаль.

Ще одиним створеним графічним елементом є DynamicComboBoxesPanel. Його особливістю є комбо-бокс меню, яке підказує можливі значення за допомогою ключових слів. Для цього тут використовується com.jidesoft.swing.AutoCompletionComboBox. Нажаль, сам по собі він здатен лише підказувати значення початок яких співпадає з введеним значенням, тобто виконувати функцію Auto-Completion. Ця особливість є не завжди зручної для роботи, і тому до текстового поля якого прикріплено MyDocumentListener, який оновлює список можливих значень після кожної зміни зробленої користувачем, виконуючи фільтрацію можливих значень.

Ще однією особливістю ComboBox елемента у Swing є те, що при зміні списку можливих значень автоматично підставляє перше значення у списку. Щоб запобігти цьому, до списка відфільтрованого списку першим номером

завжди додається текст, який набрав користувач та по якому був виконан пошук. Роботу цього механізму можна побачити на Рисунку 14.

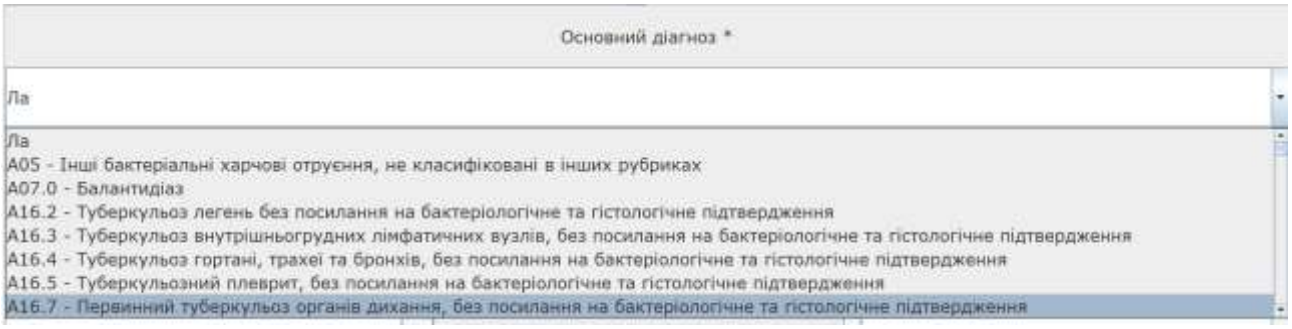


Рисунок 14 – Комбобокс з динамічним списком значень

Нові значення можна отримуються за допомогою особливого інтерфейсу SearchFunction який дозволяю сховати свою імплементаційю від інтерфейсу. Це зробленого для того щоб підтримувати збір можливих значень як із SQL серверу так і з Json файлів без змін у роботі графічного елементу.

Майже всі графічні елементи для інтерфейсу заповнення анкет розміщені за допомогою GridLayout та BorderLayout менеджерів.

Клас GridLayout представляє менеджер для розміщення елементів із заданою кількістю рядків та стовпців у прямокутній сітці. Контейнер GridLayout розділений на рівні у формі прямокутників, і кожен з компонентів розміщується у відповідному прямокутнику. Кожна комірка прямокутника має однаковий розмір, тому компонент, який вони містять, заповнює всю комірку. Коли користувач змінює або коригує розмір контейнера, розмір кожного прямокутника змінюється відповідно. При додаванні, сітка спочатку заповнюється зліва направо, а потім згори донизу.

BorderLayout - це менеджер для розміщення елементів за замовчуванням для таких віконних об'єктів, як JFrame, JWindow, JDialog, JInternalFrame тощо. BorderLayout розташовує компоненти в п'яти регіонах. Чотири сторони позначаються як північ, південь, схід та захід. Середня частина називається

центром. Кожна область може містити лише один компонент і ідентифікується відповідною константою як NORTH, SOUTH, EAST, WEST та CENTER. Майже завжди цей менджер для розміщення елементів віддає вільний простір елементу, який розташований у центрі.

3.7 Модуль WebApi для заповнення анкет на сайті

Модуль WebApi використовується для автоматичного заповнення анкет на сайті UDRG-System.com. Він приймає об’єкт типу Case та з його допомогою заповнює всі наявні значення.

Особливої уваги тут потребують елементи select з динамічно створюваними списками значень оскільки вони не дозволяються вписати значення, а лише обрати його зі списку можливих, який на момент завантаження сторінки є пустим. Для роботи з ними використовується особливий метод для заповнення під назвою setDynamicSelectValue. Суть цього методу полягає у тому що він знаходить елемент select для заповнення, імітує клік користувача мишкою на нього, після чого, у відкритому меню починає вводити значення для заповнення. У цей момент, список має скоротитися до одного елемента, на який буде імітовано клік користувача мишкою.

Через те, що частина динамічних елементів select на сайті має значну ваду – вони не можуть знайти значення якщо ввід користувача містить пробіл, введене значення скорочується, обирається лише частина від початку та до першого пробілу.

Ще одну проблему представляють поля, кількість яких може змінюватись – додаткові діагнози та процедури. Для того щоб мати можливість їх заповнювати була створена функція fillDynamicPanels яка спочатку створює необхідну кількість елементів для заповнення, після чого починає заповнювати їх введеними даними за допомогою метода setDynamicSelectValue. Результат роботи такої функції наведено на Рисунку 15.

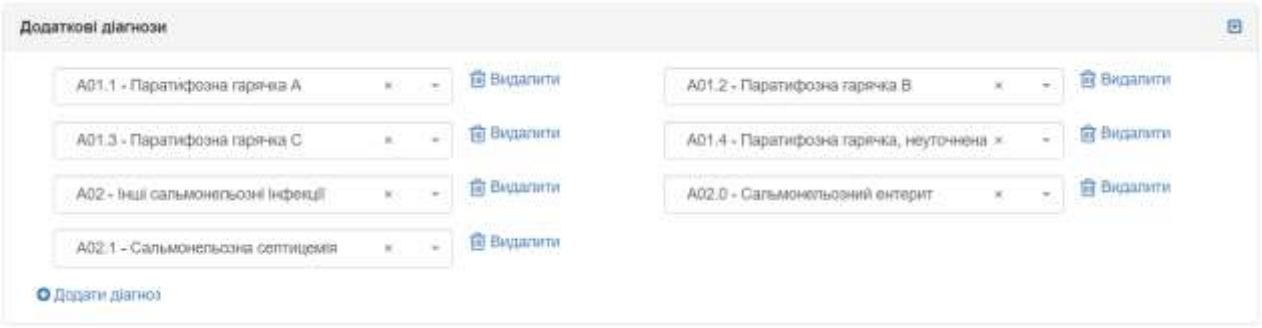


Рисунок 15 – Результат роботи функції fillDynamicPanels

ВИСНОВКИ

Вже давно відомо що системи електронного документообігу приносять багато користі під час застосування у медицині. Але створення такої системи пов’язане з певними складнощами та ризиками. Недостатня безпека зберігання даних може спричинити витік даних, що у свою чергу загрожує втратою довіри, матеріальними збитками та судовими позовами. Через це треба максимально ретельно планувати архітектуру майбутнього за стосунку і насамперед його бази даних. У той самий час, надзвичайно ускладнений інтерфейс може завади лікарям ефективно працювати з програмою.

Для розробки будо обрано мову програмування Java саме через те, що це є стандартом для розробки програмного забезпечення для закладів, де безпека даних є однією з найважливіших проблем, наприклад фінансових установ, банків. Для розробки бази даних було використано MS SQL через якісну підтримку від Microsoft та швидке усунення ризиків для безпеки.

У цій роботі біло створену автоматизовану систему для обробки даних пацієнтів лікарні яка має значно полегшити виконання своїх службових обов’язків лікарями.

1. Намперед вона пришвидшує заповнення анкет на державний сайті для розрахунку вартості медичних послуг та вагових коефіцієнтів в закладах охорони здоров’я України. Раніше лікарям постійно доводилось заповнювати анкету інформацією, яка дублюється із анкету до анкети. Розроблена система автоматизує заповнення таких полів і полегшує працю лікарів.
2. У створеній системі усунено недоліки, на які користувач може наткнутися під час заповнення анкети на сайті. Наприклад певні комбінації символів призводили до унеможливлення заповнення деяких полів.

3. Нова система також сповіщає користувача про помилки зроблені під час набору, тим самим покращуючи якість зібраних даних.
4. Іншою великою перевагою створеної системи є можливість локально зберігати відправлені дані. Це усуває необхідність у підключенні до мережі Інтернет та дозволяє заповнювати анкети незалежно від робочого стану сайту. Тобто, завдяки використанню системи, лікарі зможуть продовжувати заповнювати анкети навіть якщо на сайті проводяться технічні роботи. Єдина різниця є лише у тому що вони будуть збережені у локальній базі даних до тих пір, поки роботу сайту UDRG-System.com не буде відновлено.
5. Дублювання даних також забезпечує їх захист від зміни чи видалення у віддаленій базі даних. Раніше, після заповнення анкет, лікарі не могли перевірити чи правильно зберігаються дані та чи правильно рахується статистика. Тепер вони зможуть самостійно робити перевірку використовуючи дані, збережені у локальній базі даних.

Розроблена система має значний потенціал для подальшого розвитку. Після того як локальна база даних накопичить достатньо інформації, стане можливим проводити аналіз ефективності наданих медичних послуг. Тобто визначати як ті, чи інші процедури допомагають у лікуванні хворих. Нові лікарі тепер можуть у будь-який час подивитись історію госпіталізацій свого пацієнта, побачити які процедури призначав лікар тоді і як ефективно вони подіяли. Через подібний збір інформації можна навіть спрогнозувати чи звернеться пацієнт до лікарні повторно, як це роблять сучасні медичні клініки світу.

Таким чином була створена система, яка задовольняє потреби лікарів, дозволяє автоматично заповнювати анкети госпіталізацій хворих на сайті UDRG-System.com, зберігати заповнені анкети локально та здійснювати пошук по ключовим словам. Можливі ризики кібербезпеки вирішено через застосування локального серверу, до якого немає доступу з мережі Інтернет.

ПЕРЕЛІК ПОСИЛАНЬ

1. Медичний центр «Маунт Сінай» виграє нагороду «Девіс» за досягнення в електронній медицині [Електронний ресурс]. – 2012. – Режим доступу до ресурсу: https://www.mountsinai.org/about/newsroom/2012/the-mount-sinai-medical-center-wins-prestigious-2012-davies-award-of-excellence-for-its-electronic-medical-records-emr-system.
2. Eckel B. Thinking in Java / Bruce Eckel., 2006.
3. Повний опис бібліотеки Swing [Електронний ресурс] – Режим доступу до ресурсу: https://www.zentut.com/java-swing/ .
4. Yaseen A. Опис структури SQL серверів [Електронний ресурс] / Ahmad Yaseen – Режим доступу до ресурсу: https://www.sqlshack.com/sql-server-table-structure-overview/ .
5. Результати досліджень найкращих шрифтів для електронних ресурсів [Електронний ресурс] – Режим доступу до ресурсу: https://www.awai.com/2011/10/the-best-fonts-to-use-in-print-online-and-email/ .
6. Design Patterns: Elements of Reusable Object-Oriented Software / E.Gamma, R. Helm, R. Johnson, J. Vlissides., 1994.
7. Cormen T. Introduction to Algorithms / Thomas Cormen.. – (2009).
8. Molinaro A. SQL Cookbook: Query Solutions and Techniques for Database Developers / Anthony Molinaro. – (2009).
9. Zukowski J. The Definitive Guide to Java Swing / John Zukowski., 2000.
10. С.М. Злепко, Т.І. Овчарук, А.А. Овчарук Огляд медичних інформаційних систем. Системи обробки інформації. 2011. № 3(93). С. 189-192.
11. Bloch J. Effective Java / Joshua Bloch., 2008. – (Addison-Wesley).

12. Head First Design Patterns / E.Freeman, B. Bates, K. Sierra, E. Robson., 2004. – 694 с.
13. Goetz B. Java Concurrency in Practice / Brian Goetz., 2006. – 424 с.
14. Martin R. Clean Code: A Handbook of Agile Software Craftsmanship / Robert Martin., 2008. – 464 с.
15. Urma R. Java 8 in Action: Lambdas, Streams, and functional-style programming / R. Urma, M. Fusco, A. Mycroft., 2014. – 424 с.
16. Oaks S. Java Performance: The Definitive Guide / Scott Oaks., 2014. – 426 с.
17. Naftalin M. Java Generics and Collections / M. Naftalin, P. Wadler., 2006. – 273 с.
18. Koskela L. Test Driven: TDD and Acceptance TDD for Java Developers / Lasse Koskela., 2007. – 470 с.
19. Oaks S. Java Threads / S. Oaks, H. Wong., 1999. – 344 с.
20. Warburton R. Java 8 Lambdas: Pragmatic Functional Programming / Richard Warburton., 2014. – 182 с.
21. Martin R. Clean Architecture: A Craftsman's Guide to Software Structure and Design / Robert Martin., 2017. – 428 с.
22. Delaney K. Microsoft SQL Server 2012 Internals / K. Delaney, C. Freeman., 2013. – 982 с.
23. McKay E. UI is Communication: How to Design Intuitive, User Centered Interfaces by Focusing on Effective Communication / Everett McKay., 2013. – 378 с.
24. Lidwell W. Universal Principles of Design / W. Lidwell, K. Holden, J. Butler., 2003. – 216 с.
25. Gothelf J. Lean UX: Applying Lean Principles to Improve User Experience / J. Gothelf, J. Seiden., 2013. – 152 с.

26. Goodwin K. Designing for the Digital Age: How to Create Human-Centered Products and Services / K. Goodwin, A. Cooper., 2009. – 768 с.
27. Cocchiaro C. Selenium Framework Design in Data-Driven Testing: Build data-driven test frameworks using Selenium WebDriver, AppiumDriver, Java, and TestNG / Carl Cocchiaro., 2018. – 354 с.
28. Palani N. Advanced Selenium Web Accessibility Testing: Software Automation Testing Secrets Revealed / Narayanan Palani., 2019. – 144 с.
29. Lalou J. Apache Maven Dependency Management / Jonathan Lalou., 2013. – 160 с.
30. Krochmalski J. IntelliJ IDEA Essentials / Jaroslaw Krochmalski., 2014. – 276 с.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

СПЕЦИФІКАЦІЯ

Роботи на тему:

«Автоматизована система збирання, кодування та зберігання облікових даних ургентних та планових пацієнтів лікарні»

Виконав

Студент групи ТВз-61

Кафедри АПЕПС

Факультету ТЕФ

Сушко Дмитро Анатолійович

Київ - 2020

Позначення	Найменування	Примітки
Документація		
	Пояснювальная записка	Описується проблематика, існуючі рішення та підходи до вирішення проблем, пояснюється структура проекту та основний його функціонал.
	Керівництво користувача	Описуються інструкції по користуванню, методи вирішення найбільш поширених проблем
Компоненти		
	SqlApi	Модуль для збереження та пошуку інформації у SQL сервері за допомогою запитів.
	WebApi	Модуль заповнення анкет на веб-сторінці UDRG-System.com від імені користувача.
	CaseInput	Модуль для заповнення анкет користувачем.

М

```
public class WebApi {
//URL зайту для заповнення
 final static String URL = "https://udrg-system.com/";
//Локатори вікна для логіну
 final static By loginInput = By.name("username");
 final static By passwordInput = By.name("password");
 final static By submitButton = By.name("login");
 //Локатори кнопок на анкеті
 final static By enterNewCaseButton = By.xpath("//*[@href = '/cases/new/']");
 final static By addDiagnosisButton = By.xpath("//div[text() = ' Додати діагноз']");
 final static By addProcedureButton = By.xpath("//div[text() = ' Додати процедуру']");
//Локатори полів вводу анкети
 final static By firstNameInput = By.name("patient_name");
 final static By lastNameInput = By.name("patient_surname");
 final static By insuranceNumberInput = By.name("patient_insurance_id_number");
 final static By countryInput = By.name("patient_country");
 final static By passportDataInput = By.name("patient_passport_id");
 final static By medicalHistoryIdInput = By.name("reference_number");
 final static By sapsTwoScoreInput = By.name("saps_score_at_admission");
 final static By nhsOfficeCodeInput = By.name("nhs_office_code");
 final static By referringDoctorIdInput = By.name("referring_doctor_id");
 final static By hoursOfMechanicalVentilationInput = By.name("hours_of_mechanical_ventilation");
 final static By newbornWeightInput = By.name("newborn_weight");
 final static By sapsScoreAtDischarge = By.name("saps_score_at_discharge");
//Локатори полів з динамічно наповнюваними елементами select
 final static By additionalDiagnosesPanel = By.xpath("//div[@id = 'collapse-additional-diagnoses']//div[select]");
 final static By proceduresPanel = By.xpath("//div[@id = 'collapse-procedures']//div[select]");
//Об’єкт для роботи с браузером
 public static WebDriver driver;
//Об’єкт для гнучкого очікування певних умов
 public static WebDriverWait wait;
//Метод для ініцілазії модулю заповнення веб сторінки
 private static void initialize() {
   driver = new ChromeDriver();
   wait = new WebDriverWait(driver, 10);
 }
```

```
 protected static void register() {
  initialize();
//Перехід на сайт для заповнення анкет
  driver.get(URL);
//Розкриття браузеру на весь екран
  driver.manage().window().maximize();
//Авторизація користувачем
  wait.until(visibilityOfAllElementsLocatedBy(loginInput));
  driver.findElement(loginInput).sendKeys(login);
  driver.findElement(passwordInput).sendKeys(password);
  driver.findElement(submitButton).click();
 }
//Метод для заповнення анкети інформацією з об'єкта Case
 public static void fillCaseReport(Case someCase) {
   if (driver == null) {
    register();
  }

  wait.until(visibilityOfElementLocated(enterNewCaseButton));
  driver.findElements(enterNewCaseButton).stream().filter(WebElement::isDisplayed).findFirst().get().click();

                        wait.until(visibilityOfAllElementsLocatedBy(firstNameInput));
                        driver.findElement(firstNameInput).sendKeys(someCase.getFirstName());
                        driver.findElement(lastNameInput).sendKeys(someCase.getLastName());
  driver.findElement(birthDayDatePicker).sendKeys(someCase.getBirthDay().toWebFormat());
  new Select(driver.findElement(genderSelect)).selectByVisibleText(someCase.getGender());

  driver.findElement(insuranceNumberInput).sendKeys(someCase.getInsuranceNumber());
  driver.findElement(countryInput).sendKeys(someCase.getCountry());
  driver.findElement(passportDataInput).sendKeys(someCase.getPassportData());
  driver.findElement(medicalHistoryIdInput).sendKeys(someCase.getMedicalHistoryId());
  driver.findElement(hospitalizationDatePicker).sendKeys(someCase.getHospitalizationDate().toWebFormat());
  new Select(driver.findElement(volunteryHospitalizationTypeSelect)).selectByVisibleText(someCase.getHospitalizationVolunterity());
  new Select(driver.findElement(singleDateStatusSelect)).selectByVisibleText(someCase.getSingleDayStatus());

  new Select(driver.findElement(hospitalizationTypeSelect)).selectByVisibleText(someCase.getHospitalizationType());
```

```
 new Select(driver.findElement(admissionReasonSelect)).selectByVisibleText(someCase.getAdmissionReason());
 driver.findElement(sapsTwoScoreInput).sendKeys(someCase.getSapsTwoScore());

 setDynamicSelectValue(hospitalSelectParent, someCase.getHospital());
 driver.findElement(nhsOfficeCodeInput).sendKeys(someCase.getNhsOfficeCode());

 if (! someCase.getDoctorName().isEmpty()) {
  setDynamicSelectValue(doctorNameSelectParent, someCase.getDoctorName());
 }
 driver.findElement(referringDoctorIdInput).sendKeys(someCase.getReferringDoctor());

 setDynamicSelectValue(principalDiagnosisParent, someCase.getPrincipalDiagnosis());

 new Select(driver.findElement(typeOfCareSelect)).selectByVisibleText(someCase.getTypeOfCare());
 driver.findElement(hoursOfMechanicalVentilationInput).sendKeys(someCase.getHoursOfMechanicalVentilation());
 driver.findElement(newbornWeightInput).sendKeys(someCase.getNewbornWeight());

 driver.findElement(dischargeDatePicker).sendKeys(someCase.getDischargeDate().toWebFormat());
 new Select(driver.findElement(dischargeModeSelect)).selectByVisibleText(someCase.getDischargeMode());
 if (someCase.getDischargeDepartment() != null) {
  setDynamicSelectValue(dischargeDepartmentParent, someCase.getDischargeDepartment());
 }

 driver.findElement(sapsScoreAtDischarge).sendKeys(someCase.getSapsScoreAtDischarge());
 if (someCase.getTransferToHospital() != null) {
  setDynamicSelectValue(transferToHospitalParent, someCase.getTransferToHospital());
 }

 if (!(someCase.getAdditionalDiagnoses().size() == 1 && someCase.getAdditionalDiagnoses().get(0).isEmpty())) {
  fillDynamicPanels(additionalDiagnosesPanel, addDiagnosisButton, someCase.getAdditionalDiagnoses());
 }

 if (!(someCase.getProcedures().size() == 1 && someCase.getProcedures().get(0).isEmpty())) {
  fillDynamicPanels(proceduresPanel, addProcedureButton, someCase.getProcedures());
 }
}
(someCase);
```

```
 }

 public static void setDynamicSelectValue(By selectParent, String value) {
  driver.findElement(selectParent).click();
  driver.findElement(By.className("select2-search__field")).sendKeys(value.split(" -")[0]);
  safeClick(By.xpath("//li[text() = '" + value + "']"));
 }
//Метод імітації кліків на елемент мишкою користувачем
 public static void safeClick(By selector) {
  try {
   wait.until(visibilityOfElementLocated(selector)).click();
  }
  catch (StaleElementReferenceException e) {
   safeClick(selector);
  }
 }
//Метод для заповнення динамічно наповнюваних елементів select
 public static void fillDynamicPanels(By selector, By addElementButton, List<String> values) {
  while(getAllDisplayedElements(selector).size() < values.size()) {
   driver.findElement(addElementButton).click();
  }

  for(int i = 0; i < values.size(); i++) {
   setDynamicSelectValue(By.xpath("(" + selector.toString().replace("By.xpath: ", "") + ")[" + (i + 2) + "]"), values.get(i));
  }
 }

 public static List<WebElement> getAllDisplayedElements(By selector) {
  return driver.findElements(selector).stream()
    .filter(element -> element.isDisplayed())
    .collect(Collectors.toList());
 }
//Модуль для збору значень динамічно заповнюваних елементів select
public class GatheringOfDynamicallyPopulatedSelectElement extends WebApi {
 final static By enterNewCaseButton = By.xpath("//*[@href = '/cases/new/']");
 public static void main(String[] args) {
  if (driver == null) {
```

```
 register();
 }
 wait.until(visibilityOfElementLocated(enterNewCaseButton));
 driver.findElements(enterNewCaseButton).stream().filter(WebElement::isDisplayed).findFirst().get().click();
 while(driver.findElements(By.xpath("//*[text() = 'Завантаження інших результатів…']")).size() > 0) {
  List<WebElement> l = driver.findElements(By.xpath("//*[text() = 'Завантаження інших результатів…']"));
  if (l.isEmpty()) break;
  WebElement DIVelement = l.get(0);
  JavascriptExecutor jse = (JavascriptExecutor) driver;
  try {
   jse.executeScript("arguments[0].scrollIntoView(true)", DIVelement);
  } catch (Throwable t) {}
 }
 List<WebElement> l = driver.findElements(By.className("select2-results__option"));
 for(WebElement element : l) {
  String text = element.getText();
  System.out.print("\"" + text + "\",");
 }
}
}
//Елемент КомбоБокс для графічного інтерфейсу програми з підтримкою пошуку за частиною значення
public class DynamicComboBoxesPanel extends JPanel {
 private final AutoCompletionComboBox comboBoxList;
 private final JButton removeComboBoxButton;
 private SearchMethod searchMethod;
 public DynamicComboBoxesPanel(SearchMethod searchMethod, ActionListener actionListener) {
  this.searchMethod = searchMethod;
  setLayout(new BorderLayout());
  comboBoxList = new AutoCompletionComboBox(new String[] {});
  comboBoxList.setStrict(false);
  comboBoxList.setStrictCompletion(false);
  JTextComponent tc = (JTextComponent) comboBoxList.getEditor().getEditorComponent();
  tc.getDocument().addDocumentListener(new MyDocumentListener(comboBoxList));
  add(comboBoxList, BorderLayout.CENTER);
  removeComboBoxButton = new JButton("Видалити");
  removeComboBoxButton.addActionListener(actionListener);
  add(removeComboBoxButton, BorderLayout.EAST);
```

```
}
public String getValue() {
 return comboBoxList.getEditor().getItem().toString();
}
private class MyDocumentListener implements DocumentListener {
 AutoCompletionComboBox autoCompletionComboBox;
 private MyDocumentListener(AutoCompletionComboBox autoCompletionComboBox) {
  this.autoCompletionComboBox = autoCompletionComboBox;
 }
 @Override
 public void insertUpdate(DocumentEvent e) {
  update();
 }
 @Override
 public void removeUpdate(DocumentEvent e) {
  update();
 }
 @Override
 public void changedUpdate(DocumentEvent e) {
 }
 public void update(){
  SwingUtilities.invokeLater(() -> {
   autoCompletionComboBox.setModel(new
DefaultComboBoxModel<>(searchMethod.getAllPossibleValues(autoCompletionComboBox.getEditor().getItem().toString())));
    if (autoCompletionComboBox.getModel().getSize() != 1) {
     autoCompletionComboBox.showPopup();
    }
   });
 }
}
public interface SearchMethod {
 String[] getAllPossibleValues(String template);
}
}
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

Опис програмного модуля

Роботи на тему:

«Автоматизована система збирання, кодування та зберігання облікових даних ургентних та планових пацієнтів лікарні»

Виконав

Студент групи ТВз-61

Кафедри АПЕПС

Факультету ТЕФ

Сушко Дмитро Анатолійович

Київ – 2020

Аннотація

Програма розроблена мовою програмування Java та здатна автоматизовано заповнювати анкети госпіталізації пацієнтів на сайті UDRG-System.com використовуючи ввод користувача. Також ця система робить зручнішим введення даних оператором та зберігає дані у локальному SQL сервері. Має графічний інтерфейс, систему авторизації та контролю доступу користувачів, та перевірку введених даних на помилки.

Ключові слова: Автоматизована система обробки медичних даних, Java, Microsoft SQL Server, Swing, Selenium.

Зміст

Загальні відомості

Програма називається Hospital Management Helper, та представляє собою jar-архів які може бути запущеним на будь-якій системі де встановлена Java Virtual Machine. Також ця система потребує встановленого та налагодженого Microsoft SQL Server. Всі інші, необхідні для роботи файли та бібліотеки знаходяться всередині цього jar-архіву. Програма заповнює анкет госпіталізації на сайті UDRG-System.com та одночасно дублює їх у локальній базі даних.

Функціональне призначення

Дана система використовується для пришвидшення роботи лікарів під час заповнення електронної документації та для надання їм інформації про раніше заповненні електронні документи. Програма вміє автоматично заповнювати анкети на сайті UDRG-System.com та зберігати заповнені анкети локально. Всі внесенні дані додатково перевіряються на помилки. У будь-який момент є можливим провести пошук по ключовим словам серед попередньо збережених анкет. На даний момент аналогів цієї системи в Україні немає.

Система призначається для використання медичним персоналом, задля полегшення та пришвидшення його роботи.

Опис логічної структури

Програма розділена на три великих модуля, а саме SqlApi, WebApi та GUI.

- SqlApi відповідає за роботу з базою даних, внесення нових даних та пошук по існуючим.
- WebApi взаємодіє з сайтом UDRG-System.com та вміє виконувати функцію логіну та вносити дані про госпіталізації пацієнтів.
- GUI відповідає за графічний інтерфейс користувача та вміє передавати внесені користувачем дані до SqlApi та WebApi модулів або виводити дані отримані від них.

Також у даній системі існує механізм авторизації, який регулює доступ користувачів до чутливої інформації.

Використовувані технічні засоби, виклик і завантаження

Для роботи програми необхідний встановлений та налагоджений Microsoft SQL Server і тому саме він зумовлєю системні вимоги. Також для запуска самої системи необхідна встановленя Java Virtual Machine. Сама система розроблена таким чином, щоб використовувати незначні системні ресурси та працювати навіть на комп’ютерах з обмеженими можливостями. Інтерфейс програми адаптивний та працює на будь-який розмірах екрану монітору або ноутбуку.

Вхідні та вихідні дані

Вхідні дані вносятся користувачем під час роботи з системою. Ці дані оброблюються та використуваються у двух задачах. Перша – це заповнення анкет госпіталізацій на сайті UDRG-System.com. Друга – це збереження даних у локальній базі даних з метою подальшого їх використання для аналізу. Внесені користувачем дані перевіряються на помилки та у разі їх знаходшення, подальше використання даних не відбувається, а користувач отримує відповідне повідомлення про помилку. Коли помилка усунена, після перевірки дані будут використані для заповнення та збереження. Без авторизації під час початку роботи з системою, користувач не отримує доступу для роботи з даними.